



BEA WebLogic Workshop™ Help

Version 8.1 SP2
November 2003

Copyright

Copyright © 2003 BEA Systems, Inc. All Rights Reserved.

Restricted Rights Legend

This software and documentation is subject to and made available only pursuant to the terms of the BEA Systems License Agreement and may be used or copied only in accordance with the terms of that agreement. It is against the law to copy the software except as specifically allowed in the agreement. This document may not, in whole or in part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine readable form without prior consent, in writing, from BEA Systems, Inc.

Use, duplication or disclosure by the U.S. Government is subject to restrictions set forth in the BEA Systems License Agreement and in subparagraph (c)(1) of the Commercial Computer Software–Restricted Rights Clause at FAR 52.227–19; subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227–7013, subparagraph (d) of the Commercial Computer Software—Licensing clause at NASA FAR supplement 16–52.227–86; or their equivalent.

Information in this document is subject to change without notice and does not represent a commitment on the part of BEA Systems. THE SOFTWARE AND DOCUMENTATION ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. FURTHER, BEA Systems DOES NOT WARRANT, GUARANTEE, OR MAKE ANY REPRESENTATIONS REGARDING THE USE, OR THE RESULTS OF THE USE, OF THE SOFTWARE OR WRITTEN MATERIAL IN TERMS OF CORRECTNESS, ACCURACY, RELIABILITY, OR OTHERWISE.

Trademarks or Service Marks

BEA, Jolt, Tuxedo, and WebLogic are registered trademarks of BEA Systems, Inc. BEA Builder, BEA Campaign Manager for WebLogic, BEA eLink, BEA Liquid Data for WebLogic, BEA Manager, BEA WebLogic Commerce Server, BEA WebLogic Enterprise, BEA WebLogic Enterprise Platform, BEA WebLogic Enterprise Security, BEA WebLogic Express, BEA WebLogic Integration, BEA WebLogic Personalization Server, BEA WebLogic Platform, BEA WebLogic Portal, BEA WebLogic Server, BEA WebLogic Workshop and How Business Becomes E–Business are trademarks of BEA Systems, Inc.

All other trademarks are the property of their respective companies.

Table of Contents

WebServices Samples.....	1
 async Samples.....	2
 Buffer.jws Sample.....	3
 Conversation.jws Sample.....	5
 HelloWorldAsync.jws Sample.....	8
 HelloWorldAsyncControl.jcx Sample.....	10
 controlFactory Samples.....	15
 ServiceFactoryClient.jws Sample.....	16
 SlowService.jws Sample.....	19
 SlowServiceControl.jcx Sample.....	21
 Controller.jpf Sample.....	26
 controlProjectTest Samples.....	27
 ControlTest.jws Sample.....	28
 creditReport Samples.....	31
 Bank.jws Sample.....	32
 BankControl.jcx Sample.....	34
 CreditReport.jws Sample.....	40
 IRS.jws Sample.....	46
 IRSControl.jcx Sample.....	48
 database Samples.....	54
 customer_db Samples.....	55
 Customer.java Sample.....	56
 CustomerDB.jcx Sample.....	59

Table of Contents

CustomerDBClient.jws Sample.....	63
dynamicSQL Samples.....	69
DynamicSQL.jws Sample.....	70
ItemsDBControl.jcx Sample.....	72
lucky_number_db Samples.....	74
LuckyNumber.jws Sample.....	75
LuckyNumberControl.jcx Sample.....	76
LuckyNumberDB.jcx Sample.....	79
LuckyNumberDBClient.jws Sample.....	81
PlayerRecord.java Sample.....	84
xmlBean Samples.....	85
CustomerDB_XMLBean.jcx Sample.....	86
CustomerDB_XMLBeanClient.jws Sample.....	88
ItemsDB_XMLBean.jcx Sample.....	91
ItemsDB_XMLBean_Client.jws Sample.....	92
ejbControl Samples.....	94
AccountEJBClient.jws Sample.....	95
AccountEJBControl.jcx Sample.....	99
TraderEJBClient.jws Sample.....	100
TraderEJBControl.jcx Sample.....	102
Error.jsp Sample.....	103
HelloWorld.jws Sample.....	104
Index.jsp Sample.....	105

Table of Contents

interop Samples.....	112
dotNET Samples.....	113
ConversationClient.asmx.cs Sample.....	114
Readme.html Sample.....	120
jms Samples.....	122
AccountPublish.jws Sample.....	123
AccountPublishJMSControl.jcx Sample.....	126
AccountSubscribe.jws Sample.....	128
AccountSubscribeJMSControl.jcx Sample.....	132
CustomJMSClient.jws Sample.....	134
CustomJMSControl.jcx Sample.....	136
SimpleJMS.jws Sample.....	138
SimpleJMSControl.jcx Sample.....	140
jms_xmlProtocol Samples.....	141
ClientJWS.jws Sample.....	142
ClientXML.java Sample.....	143
JMS_XMLProtocol.jws Sample.....	145
localControls Samples.....	146
asynch Samples.....	147
BufferJC.java Sample.....	148
BufferJCImpl.jcs Sample.....	149
basics Samples.....	151
Hello.java Sample.....	152

Table of Contents

HelloImpl.jcs Sample.....	153
LocalControlTest.jws Sample.....	155
nestedControls Samples.....	159
CustomerAccountEJB.jcx Sample.....	160
ItemsDatabase.jcx Sample.....	161
poUtil Samples.....	163
POUtil.java Sample.....	164
POUtilImpl.jcs Sample.....	165
VerifyFunds.java Sample.....	166
VerifyFundsImpl.jcs Sample.....	167
tags Samples.....	171
ItemsDatabaseControl.jcx Sample.....	172
POVerify-tags.xml Sample.....	173
POVerify.java Sample.....	174
POVerifyImpl.jcs Sample.....	175
proxy Samples.....	178
mazegen Samples.....	179
MazeGenerator.jws Sample.....	180
MazeMap.jsx Sample.....	185
Partition.java Sample.....	187
register Samples.....	189
Contact.java Sample.....	190
Person.java Sample.....	191

Table of Contents

RegisterPerson.jws Sample.....	192
security Samples.....	196
roleBased Samples.....	197
Bank.jws Sample.....	198
BankControl.jcx Sample.....	199
createUser Samples.....	204
CreateUser.jws Sample.....	205
VeriCheck.jws Sample.....	208
transport Samples.....	210
basicAuthentication Samples.....	211
BasicAuthentication.jws Sample.....	212
clientCert Samples.....	213
WebServiceA.jws Sample.....	214
WebServiceB.jws Sample.....	216
WebServiceBControl.jcx Sample.....	217
helloWorldSecure Samples.....	222
HelloWorldSecure.jws Sample.....	223
HelloWorldSecureClient.jws Sample.....	224
HelloWorldSecureControl.jcx Sample.....	225
wsse Samples.....	228
callback Samples.....	229
client Samples.....	230
Client.jws Sample.....	231

Table of Contents

TargetControl.jcx Sample.....	232
TargetControlPolicy.wsse Sample.....	237
Readme.html Sample.....	238
target Samples.....	239
Target.jws Sample.....	240
TargetPolicy.wsse Sample.....	241
reqResp Samples.....	242
client Samples.....	243
Client.jws Sample.....	244
MyCompanyControl.jcx Sample.....	245
MyCompanyControlPolicy.wsse Sample.....	248
mycompany Samples.....	250
MyCompany.jws Sample.....	251
MyCompanyContract.wsdl Sample.....	252
MyCompanySecurityPolicy.wsse Sample.....	254
Readme.html Sample.....	256
usertoken Samples.....	257
Readme.html Sample.....	258
webServiceA Samples.....	259
WebServiceA.jws Sample.....	260
WebServiceBControl.jcx Sample.....	261
WebServiceBControlPolicy.wsse Sample.....	264
webServiceB Samples.....	265

Table of Contents

PolicyB.wsse Sample.....	266
WebServiceB.jws Sample.....	267
WebServiceB.wsdl Sample.....	268
service Samples.....	270
QuoteClient.jws Sample.....	271
QuoteService.jws Sample.....	272
QuoteServiceControl.jcx Sample.....	274
timer Samples.....	279
AdvancedTimer.jws Sample.....	280
LegacySystem.jws Sample.....	283
LegacySystemControl.jcx Sample.....	285
SimpleTimer.jws Sample.....	291
xmlBeans Samples.....	294
cursor Samples.....	295
BatchOrderSimple.xml Sample.....	296
InventoryItem.xml Sample.....	297
MixedContent.jws Sample.....	298
TokenTypes.jws Sample.....	302
schema Samples.....	305
PriceSummary.xml Sample.....	306
PurchaseOrder.xml Sample.....	307
PurchaseOrderSimple.xml Sample.....	308
SchemaChoice.jws Sample.....	309

Table of Contents

SchemaEnum.jws Sample.....	312
SimpleAccess.jws Sample.....	316
ThresholdEnums.java Sample.....	320
ThresholdEnumsImpl.jcs Sample.....	321
ThresholdService.jws Sample.....	324
TypeHierarchyPrinter.java Sample.....	326
TypeHierarchyPrinterService.jws Sample.....	330
XsdConfig.jws Sample.....	332
xquery Samples.....	334
Employees.xml Sample.....	335
SelectPath.jws Sample.....	337
SimpleExpressions.jws Sample.....	342
xqueryMap Samples.....	347
InputMapMultiple.jws Sample.....	348
InputMapMultipleTest.xml Sample.....	350
OutputMap.jws Sample.....	351
OutputScriptMap.jws Sample.....	353
Person.java Sample.....	354
PersonScript.jsx Sample.....	355
SimpleMap.jws Sample.....	356

WebServices Samples

This section contains source code for the following samples.

Samples Included in This Section

async Samples

controlFactory Samples

Controller.jspf Sample

controlProjectTest Samples

creditReport Samples

database Samples

ejbControl Samples

Error.jsp Sample

HelloWorld.jws Sample

Index.jsp Sample

interop Samples

jms Samples

jms_xmlProtocol Samples

localControls Samples

proxy Samples

security Samples

service Samples

timer Samples

xmlBeans Samples

xqueryMap Samples

async Samples

This section contains source code for the following samples.

Samples Included in This Section

Buffer.jws Sample

Conversation.jws Sample

HelloWorldAsync.jws Sample

HelloWorldAsyncControl.jcx Sample

Buffer.jws Sample

This topic includes the source code for the Buffer.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/async/

Sample Source Code

```
package async;

import com.bea.control.JwsContext;

/**
 * <p>A web service that demonstrates the @jws:buffer tag to queue high-traffic requests.</p>
 *
 * <p>Uses a TimerControl to delay sending a response back to the client, simulating waiting
 * for a slow back-end service to respond to requests from this web service. The start method
 * is buffered, in order to accept many requests without blocking.</p>
 * @common:target-namespace namespace="http://workshop.bea.com/Buffer"
 */

public class Buffer implements com.bea.jws.WebService
{
    /** @common:context */
    JwsContext context;

    /**
     * @jc:timer timeout="10 seconds"
     * @common:control
     */
    private com.bea.control.TimerControl delayTimer;

    /**
     * <p>the Callback interface is the definition of which messages the client will
     * accept from us via the callback variable.</p>
     */
    public interface Callback
    {
        /**
         * <p>BufferResult is the message we will send back to the client when the result for
         * that client is ready.</p>
         * @jws:conversation phase="finish"
         */
        public void BufferResult(String hello);
    }

    /**
     * <p>callback is the variable that represents the client connection. It's used
     * to enable our methods and events to send messages back to the client.</p>
     */
    public Callback callback;
```

navWebServices.html Sample

```
/**
 * <p>Starts the client interaction with Buffer.jws.</p>
 *
 * <p>It is difficult to demonstrate using the Test View and a simple web service, but
 * the @common:message-buffer tag causes WebLogic Server to implement a message queue for
 * each method that is marked with @common:message-buffer. Requests to this method are the
 * automatically queued and are fed to the web service as resources allow. This can
 * vastly increase the load that your web service is able to handle.</p>
 *
 * <p>The client sends this, and some time later Buffer replies by calling BufferResult.
 * Because this is an ongoing transaction, startBufferAsync is marked as a
 * "conversation start" method. This means the system will track which clients
 * have called us, and where to send the result for each. This is known as
 * <i>correlation</i>.</p>
 *
 * @common:operation
 * @jws:conversation phase="start"
 * @common:message-buffer enable="true"
 */
public void startBufferAsync()
{
    // all we do here is start the timer.
    delayTimer.start();
}

/**
 * <p>The handler for delayTimer's onTimeout event.</p>
 *
 * <p>When the timer expires, this code will be run
 * to send the result back to the client who asked for it.</p>
 */
private void delayTimer_onTimeout(long time)
{
    // send the client a hello message.
    callback.BufferResult("Hello, asynchronous world!");

    // we don't want any more timer events for this conversation.
    delayTimer.stop();

    // this conversation is over, we don't need to remember anything about the client anymore
    // finish() is a method of Buffer's superclass, ServiceControl.
    context.finishConversation();
}
}
```

Conversation.jws Sample

This topic includes the source code for the Conversation.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/async/

Sample Source Code

```
package async;

import com.bea.control.JwsContext;

/**
 * <p>Demonstrates use of the @jws:conversation tag to manage the lifetime of the service
 * and provide data persistence and message correlation.</p>
 *
 * <p>Remember that multiple clients might invoke a web service simultaneously. When the
 * web service stores data relevant to the client or calls additional services
 * in order to process a client's request, the service must be able to process returned
 * data from the external services in the context of the specific client it relates
 * to. This is all automatic when conversations are used.</p>
 *
 * <p>This service uses an external service that is asynchronous, but exposes polling
 * methods in addition to callbacks so that a client that can't accept callbacks
 * can use it.</p>
 *
 * <p>Remember that not all clients are capable of accepting callbacks. Specifically,
 * clients operating from behind firewalls may not be able to receive asynchronous
 * callbacks. You may wish to provide a synchronous interface, like this one,
 * for such clients. If a client can accept callbacks, it must send a callbackURL
 * as part of any "start conversation" method invocation. The callbackURL is
 * available via the getCallbackLocation method of the JwsContext interface,
 * which is available in all WebLogic web services.</p>
 *
 * <p>To see the behavior in the Test View, invoke startRequest and then getRequestStatus
 * several times quickly. Examine the results of the calls to testHelloAsync.
 * The result of getRequestStatus will change once the timer in HelloWorldAsync
 * has expired.</p>
 * @common:target-namespace namespace="http://workshop.bea.com/Conversation"
 */
public class Conversation implements com.bea.jws.WebService
{
    /**
     * @common:control
     */
    private HelloWorldAsyncControl helloAsync;

    /**
     * @common:context
     */
    JwsContext context;

    /**
```

navWebServices.html Sample

```
* <p>This variable holds what the HelloWorldAsync service has sent back, if anything.
* By designating the methods of this web service as part of a conversation
* with the @common:conversation tags, member variables of this web service
* automatically become persistent for the duration of the conversation.</p>
*/
private String helloAsyncReturn;

/**
 * <p>This variable holds the callbackURL of the client. It is obtained from the
 * JwsContext interface during the "start" method of the conversation. If the client
 * does not supply a callback URL, then callbacks cannot be sent to the client.
 *
 * In this web service, we'll assume that if the client can't accept callbacks it
 * will use the polling interface (getRequestStatus) to get the current state and
 * it will use terminateRequest when the client is done using this web service.</p>
 */
private String callbackURL = null;

public Callback callback;

public interface Callback
{
    /**
     * Callback to invoke on the client when the external service
     * returns its result. Will only be called if the client can
     * accept callbacks and told us where to send them.
     *
     * If this callback is used, it implicitly terminates the
     * conversation with no action required on the part of the
     * client.
     *
     * @jws:conversation phase="finish"
     */
    public void onResultReady(String result);
}

/**
 * <p>Starts the conversation and makes a request to the HelloWorldAsync service.
 * To continue the sample after calling this method, click the conversation ID in the
 * log, then select getRequestStatus until the request is fulfilled. Then click
 * terminateRequest.</p>
 *
 * <p>Pass in <b>false</b> for useCallbacks to simulate a client that cannot accept
 * callbacks.</p>
 *
 * <p>Pass in <b>true</b> for useCallbacks to allow callbacks.</p>
 *
 * @common:operation
 * @jws:conversation phase="start"
 */
public void startRequest(boolean useCallbacks)
{
    /**
     * The servlet that underlies Test View can receive callbacks, so
     * we can't demonstrate a client that cannot accept callbacks unless
     * we provide an explicit way for the client to say not to use them.
     * If you specify 'false' for useCallbacks in Test View when using this
     * web service, it will behave as though the client cannot accept
     * callbacks.
     */
}
```


navWebServices.html Sample

```
        */
        if( useCallbacks ) {
            callbackURL = context.getCallbackLocation();
        }
        // start the HelloWorldAsync service.
        helloAsync.HelloAsync();
    }

    /**
     * <p>getRequestStatus is used to poll Conversation.jws to determine whether
     * we've heard back from HelloWorldAsync.</p>
     *
     * @common:operation
     * @jws:conversation phase="continue"
     */
    public String getRequestStatus()
    {
        if (helloAsyncReturn == null)
            return "HelloAsync has not yet returned it's message";
        else
            return "HelloAsync said: '" + helloAsyncReturn + "'";
    }

    /**
     * <p>Used to tell Conversation.jws that the current conversation is
     * no longer needed.</p>
     *
     * @common:operation
     * @jws:conversation phase="finish"
     */
    public void terminateRequest()
    { }

    /**
     * <p>Handler for HelloWorldAsync's HelloResult callback.</p>
     *
     * <p>Store HelloWorldAsync's response in a member variable so that future calls to
     * testHelloAsync will give the current status.</p>
     */
    public void helloAsync_onHelloResult(java.lang.String hello)
    {
        helloAsyncReturn = hello;

        /*
         * If the client can accept callbacks, use the callback to send the result.
         * Since onResultReady is marked phase="finish", the conversation will be
         * terminated after this call with no further action required from the
         * client.
         */
        if( callbackURL != null ) {
            callback.onResultReady(hello);
        }
    }
}
```

HelloWorldAsync.jws Sample

This topic includes the source code for the HelloWorldAsync.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/async/

Sample Source Code

```
package async;

/**
 * <p> A web service that uses a TimerControl to delay sending a response back to the client.</p>
 *
 * <p>The timer simulates waiting for a slow back end service to complete work for us. We
 * use a callback to asynchronously notify the client when the simulated operation is
 * complete.</p>
 * @common:target-namespace namespace="http://workshop.bea.com/HelloWorldAsync"
 */
public class HelloWorldAsync implements com.bea.jws.WebService
{
    /**
     * @jc:timer timeout="10 seconds"
     * @common:control
     */
    private com.bea.control.TimerControl helloDelay;

    /**
     * <p>callback is the variable that represents the client connection. It's used
     * to enable our service to send messages back to the client asynchronously.
     * callback is type Callback, which is defined next;</p>
     */
    public Callback callback;

    /**
     * <p>the Callback interface is the definition of which messages the client will
     * accept from us via the callback variable.</p>
     */
    public interface Callback
    {
        /**
         * <p>HelloResult is the message we will send back to the client some time after
         * the client initiates the operation.</p>
         *
         * <p>We mark the callback as finishing the conversation because once we
         * invoke the callback the interaction between client and this service is
         * complete. If we didn't use the @common:conversation tag here to finish
         * the conversation, we would have to call finishConversation() on this
         * service's context object.</p>
         *
         * @jws:conversation phase="finish"
         */
        public void onHelloResult(String hello);
    }
}
```

navWebServices.html Sample

```
/**
 * <p>The client starts the interaction by calling HelloAsync.</p>
 *
 * <p>The client sends this, and some time later our service replies by calling
 * callback.onHelloResult.</p>
 *
 * <p>Because we need to remember which client called us, this is a "conversation start"
 * method. This means the system will automatically track which clients have called
 * us, and where to send the result for each. This is known as <i>correlation</i>.</p>
 *
 * @common:operation
 * @jws:conversation phase="start"
 */
public void HelloAsync()
{
    // all we do here is start the timer.
    helloDelay.start();
    return;
}

/**
 * <p>The handler for helloTimer's onTimeout event.</p>
 *
 * <p>When the timer expires, this method will be invoked. When that
 * happens, we'll send the result back to the client who asked for it.</p>
 */
private void helloDelay_onTimeout(long time)
{
    // send the client a hello message.
    callback.onHelloResult("Hello, asynchronous world");

    // we don't want any more timer events for this conversation.
    helloDelay.stop();

    return;
}
}
```

HelloWorldAsyncControl.jcx Sample

This topic includes the source code for the HelloWorldAsyncControl.jcx Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/async/

Sample Source Code

```
package async;

/**
 * <p> A web service that uses a TimerControl to delay sending a response back to the client.</p>
 * @jc:location http-url="HelloWorldAsync.jws" jms-url="HelloWorldAsync.jws"
 * @jc:wsdl file="#HelloWorldAsyncWsd1"
 * @editor-info:link autogen-style="java" source="HelloWorldAsync.jws" autogen="true"
 */
public interface HelloWorldAsyncControl extends com.bea.control.ControlExtension, com.bea.control.CallbackControl
{
    public static class StartHeader
        implements java.io.Serializable
    {
        public java.lang.String conversationID;
        public java.lang.String callbackLocation;
    }

    public static class ContinueHeader
        implements java.io.Serializable
    {
        public java.lang.String conversationID;
    }

    public static class CallbackHeader
        implements java.io.Serializable
    {
        public java.lang.String conversationID;
    }

    public interface Callback extends com.bea.control.ServiceControl.Callback
    {
        /**
         * <p>HelloResult is the message we will send back to the client some time after the client sends this message.</p>
         * @jc:conversation phase="finish"
         */
        public void onHelloResult (java.lang.String hello);
    }

    /**
     * <p>The client starts the interaction by calling HelloAsync.</p> <p>The client sends this message when the interaction is finished.</p>
     * @jc:conversation phase="start"
     */
    public void HelloAsync ();
}
```

navWebServices.html Sample

```
static final long serialVersionUID = 1L;
}

/** @common:define name="HelloWorldAsyncWsd1" value::
<?xml version="1.0" encoding="utf-8"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:conv="http://www.openuri.org/2002/01/20/conv"
    <types>
        <s:schema elementFormDefault="qualified" targetNamespace="http://workshop.bea.com/HelloWorldAsyncWsd1">
            <s:element name="onHelloResultResponse">
                <s:complexType>
                    <s:sequence/>
                </s:complexType>
            </s:element>
            <s:element name="onHelloResult">
                <s:complexType>
                    <s:sequence>
                        <s:element name="hello" type="s:string" minOccurs="0"/>
                    </s:sequence>
                </s:complexType>
            </s:element>
            <s:element name="HelloAsync">
                <s:complexType>
                    <s:sequence/>
                </s:complexType>
            </s:element>
            <s:element name="HelloAsyncResponse">
                <s:complexType>
                    <s:sequence/>
                </s:complexType>
            </s:element>
        </s:schema>

        <s:schema elementFormDefault="qualified" targetNamespace="http://www.openuri.org/2002/01/20/conv">
            <s:element name="StartHeader" type="conv:StartHeader"/>
            <s:element name="ContinueHeader" type="conv:ContinueHeader"/>
            <s:element name="CallbackHeader" type="conv:CallbackHeader"/>
            <s:complexType name="StartHeader">
                <s:sequence>
                    <s:element minOccurs="0" maxOccurs="1" name="conversationID" type="s:string"/>
                    <s:element minOccurs="0" maxOccurs="1" name="callbackLocation" type="s:string"/>
                </s:sequence>
            </s:complexType>
            <s:complexType name="ContinueHeader">
                <s:sequence>
                    <s:element minOccurs="1" maxOccurs="1" name="conversationID" type="s:string"/>
                </s:sequence>
            </s:complexType>
            <s:complexType name="CallbackHeader">
                <s:sequence>
                    <s:element minOccurs="1" maxOccurs="1" name="conversationID" type="s:string"/>
                </s:sequence>
            </s:complexType>
        </s:schema>
    </types>
    <message name="onHelloResultSoapIn">
        <part name="parameters" element="s0:onHelloResultResponse"/>
    </message>
    <message name="onHelloResultSoapOut">
        <part name="parameters" element="s0:onHelloResult"/>
    </message>
```

navWebServices.html Sample

```
<message name="HelloAsyncSoapIn">
  <part name="parameters" element="s0:HelloAsync"/>
</message>
<message name="HelloAsyncSoapOut">
  <part name="parameters" element="s0:HelloAsyncResponse"/>
</message>
<message name="onHelloResultHttpGetIn"/>
<message name="onHelloResultHttpGetOut">
  <part name="hello" type="s:string"/>
</message>
<message name="HelloAsyncHttpGetIn"/>
<message name="HelloAsyncHttpGetOut"/>
<message name="onHelloResultHttpPostIn"/>
<message name="onHelloResultHttpPostOut">
  <part name="hello" type="s:string"/>
</message>
<message name="HelloAsyncHttpPostIn"/>
<message name="HelloAsyncHttpPostOut"/>
<message name="StartHeader_literal">
  <part name="StartHeader" element="conv:StartHeader"/>
</message>
<message name="CallbackHeader_literal">
  <part name="CallbackHeader" element="conv:CallbackHeader"/>
</message>
<portType name="HelloWorldAsyncSoap">
  <operation name="onHelloResult">
    <documentation><p>HelloResult is the message we will send back to the client so</p></documentation>
    <output message="s0:onHelloResultSoapOut"/>
    <input message="s0:onHelloResultSoapIn"/>
  </operation>
  <operation name="HelloAsync">
    <documentation><p>The client starts the interaction by calling HelloAsync.</p></documentation>
    <input message="s0:HelloAsyncSoapIn"/>
    <output message="s0:HelloAsyncSoapOut"/>
  </operation>
</portType>
<portType name="HelloWorldAsyncHttpGet">
  <operation name="onHelloResult">
    <documentation><p>HelloResult is the message we will send back to the client so</p></documentation>
    <output message="s0:onHelloResultHttpGetOut"/>
    <input message="s0:onHelloResultHttpGetIn"/>
  </operation>
  <operation name="HelloAsync">
    <documentation><p>The client starts the interaction by calling HelloAsync.</p></documentation>
    <input message="s0:HelloAsyncHttpGetIn"/>
    <output message="s0:HelloAsyncHttpGetOut"/>
  </operation>
</portType>
<portType name="HelloWorldAsyncHttpPost">
  <operation name="onHelloResult">
    <documentation><p>HelloResult is the message we will send back to the client so</p></documentation>
    <output message="s0:onHelloResultHttpPostOut"/>
    <input message="s0:onHelloResultHttpPostIn"/>
  </operation>
  <operation name="HelloAsync">
    <documentation><p>The client starts the interaction by calling HelloAsync.</p></documentation>
    <input message="s0:HelloAsyncHttpPostIn"/>
    <output message="s0:HelloAsyncHttpPostOut"/>
  </operation>
</portType>
<binding name="HelloWorldAsyncSoap" type="s0:HelloWorldAsyncSoap">
```

navWebServices.html Sample

```
<soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
<operation name="onHelloResult">
  <soap:operation soapAction="http://workshop.bea.com/HelloWorldAsync/onHelloResult" style="document"/>
  <cw:transition phase="finish"/>
  <input>
    <soap:body use="literal"/>
  </input>
  <output>
    <soap:body use="literal"/>
    <soap:header wsdl:required="true" message="s0:CallbackHeader_literal" part="CallbackHeader"/>
  </output>
</operation>
<operation name="HelloAsync">
  <soap:operation soapAction="http://workshop.bea.com/HelloWorldAsync/HelloAsync" style="document"/>
  <cw:transition phase="start"/>
  <input>
    <soap:body use="literal"/>
    <soap:header wsdl:required="true" message="s0:StartHeader_literal" part="StartHeader"/>
  </input>
  <output>
    <soap:body use="literal"/>
  </output>
</operation>
</binding>
<binding name="HelloWorldAsyncHttpGet" type="s0:HelloWorldAsyncHttpGet">
  <http:binding verb="GET"/>
  <operation name="onHelloResult">
    <http:operation location="/onHelloResult"/>
    <cw:transition phase="finish"/>
    <input>
      <mime:mimeXml part="Body"/>
    </input>
    <output>
      <http:urlEncoded/>
    </output>
  </operation>
  <operation name="HelloAsync">
    <http:operation location="/HelloAsync"/>
    <cw:transition phase="start"/>
    <input>
      <http:urlEncoded/>
    </input>
    <output/>
  </operation>
</binding>
<binding name="HelloWorldAsyncHttpPost" type="s0:HelloWorldAsyncHttpPost">
  <http:binding verb="POST"/>
  <operation name="onHelloResult">
    <http:operation location="/onHelloResult"/>
    <cw:transition phase="finish"/>
    <input>
      <mime:mimeXml part="Body"/>
    </input>
    <output>
      <mime:content type="application/x-www-form-urlencoded"/>
    </output>
  </operation>
  <operation name="HelloAsync">
    <http:operation location="/HelloAsync"/>
    <cw:transition phase="start"/>
    <input>
```

navWebServices.html Sample

```
<mime:content type="application/x-www-form-urlencoded"/>
</input>
<output/>
</operation>
</binding>
<service name="HelloWorldAsync">
  <documentation>&lt;p> A web service that uses a TimerControl to delay sending a resp
  <port name="HelloWorldAsyncSoap" binding="s0:HelloWorldAsyncSoap">
    <soap:address location="http://localhost:7001/async/HelloWorldAsync.jws"/>
  </port>
  <port name="HelloWorldAsyncHttpGet" binding="s0:HelloWorldAsyncHttpGet">
    <http:address location="http://localhost:7001/async/HelloWorldAsync.jws"/>
  </port>
  <port name="HelloWorldAsyncHttpPost" binding="s0:HelloWorldAsyncHttpPost">
    <http:address location="http://localhost:7001/async/HelloWorldAsync.jws"/>
  </port>
</service>
</definitions>
* ::
*/
```


controlFactory Samples

This section contains source code for the following samples.

Samples Included in This Section

ServiceFactoryClient.jws Sample

SlowService.jws Sample

SlowServiceControl.jcx Sample

ServiceFactoryClient.jws Sample

This topic includes the source code for the ServiceFactoryClient.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/controlFactory/

Sample Source Code

```
package controlFactory;

import java.util.HashMap;
import java.io.Serializable;

/**
 * <p>Demonstrates use of a control factory.</p>
 *
 * <p>A control factory is used to dynamically create multiple
 * instances of a Service control. When all instances of the
 * Service control have returned results, a callback is sent
 * to the client listing the result of each Service control.</p>
 * @common:xmlns namespace="http://openuri.org/bea/samples/workshop/controlFactory" prefix="ns0"
 * @common:target-namespace namespace="http://workshop.bea.com/ServiceFactoryClient"
 */
public class ServiceFactoryClient implements com.bea.jws.WebService
{
    /**
     * @common:control
     */
    private controlFactory.SlowServiceControlFactory m_services;

    int m_numServices;
    int m_numServicesReturned;
    HashMap m_serviceHash;
    Result [] m_results;

    /**
     * A utility class used to collect and return results
     */
    public static class Result implements Serializable
    {
        public String name;
        public long time;

        public Result(){};
    }

    public Callback callback;

    public interface Callback
    {
        /**
         * Callback used to return results to the client. An array
         * of Result objects is serialized into the outgoing XML

```

navWebServices.html Sample

```

* message using an XQuery map.
*
* @jws:conversation phase="finish"
* @jws:parameter-xml schema-element="ns0:completeServices" xquery::
* declare namespace ns0="http://workshop.bea.com/ServiceFactoryClient"
* declare namespace ns1="http://openuri.org/bea/samples/workshop/controlFactory"
*
* let $i := $input/ns0:result
* return
*     <ns1:completeServices>
*         <ns1:result>
*             {for $r in $i/ns0:Result
*             return
*                 <ns1:service>
*                     <ns1:name>{data($r/ns0:name)}</ns1:name>
*                     <ns1:time>{data($r/ns0:time)}</ns1:time>
*                 </ns1:service>}
*             </ns1:result>
*         </ns1:completeServices>:::
*/
public void allServicesComplete(Result [] result);
}

/**
* <p>Launch the number of instances of the service control that
* are specified by <b>numServices</b>.</p>
*
* @common:operation
* @jws:conversation phase="start"
*/
public void startServices(int numServices)
{
    int i;

    if(numServices > 0)
    {
        m_numServices = numServices;
        m_numServicesReturned = 0;
        m_serviceHash = new HashMap();
        m_results = new Result[numServices];

        for(i=0; i<numServices; i++)
        {
            String serviceName = "Service" + i;

            // use the control factory's create method to create a control instance
            controlFactory.SlowServiceControl service = m_services.create();

            // use the control like you would any other control
            service.requestInfo(serviceName);

            // stash the time each control is launched, indexed by instance name
            m_serviceHash.put(serviceName, new Long((new java.util.Date().getTime())));
        }
    }
}

/**
* <p>Callback handler for the control's infoReady callback.</p>

```

navWebServices.html Sample

```
*
* <p>For a control factory, callback handlers take an extra argument that is
* the control instance, so that the handler code can tell which control is
* sending the callback.</p>
*/
private void m_services_infoReady(SlowServiceControl service, String name)
{
    // compute how many seconds since this control was launched
    long timeTaken = (new java.util.Date().getTime() - ((Long)m_serviceHash.get(name)).longValue());

    // make a new Result object and populate it with control name and time taken
    m_results[m_numServicesReturned] = new Result();
    m_results[m_numServicesReturned].name = name;
    m_results[m_numServicesReturned].time = timeTaken;

    // keep track of how many services have returned
    m_numServicesReturned++;

    // when all services have returned, tell the client via the callback
    if(m_numServicesReturned == m_numServices)
    {
        callback.allServicesComplete(m_results);
    }
}
}
```

SlowService.jws Sample

This topic includes the source code for the SlowService.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/controlFactory/

Sample Source Code

```
package controlFactory;

/**
 * <p>A dummy service that represents a back end service
 * that takes an psuedo-random amount of time to do its
 * work.</p>
 * @common:target-namespace namespace="http://workshop.bea.com/SlowService"
 */

public class SlowService implements com.bea.jws.WebService
{
    /**
     * @jc:timer timeout="5 seconds"
     * @common:control
     */
    private com.bea.control.TimerControl delayTimer;

    String m_sName;

    public Callback callback;

    public interface Callback
    {
        /**
         * Callback that is sent to the client when this
         * service is done doing its work.
         *
         * @jws:conversation phase="finish"
         */
        public void infoReady(String name);
    }

    /**
     * <p>The request from the client to do some work. After some
     * amount of time, the infoReady callback will be sent to the
     * client.</p>
     *
     * @common:operation
     * @jws:conversation phase="start"
     */
    public long requestInfo(String name)
    {
        m_sName = name;
    }
}
```

navWebServices.html Sample

```
// compute a random delay between 5 and 15 seconds
long delay = 5 + (long)(Math.random() * 10.0);

// set the timeout on the timer control.
// this overrides the value set with the @jc:timer tag.
delayTimer.setTimeout(delay);

// start the timer
delayTimer.start();

// return the time it will take for this service to do its work
return delay;
}

/**
 * Callback handler for the timer control's onTimeout
 * callback. This indicates that the work this service
 * does is complete and the client should be notified
 * via the infoReady callback.
 */
private void delayTimer_onTimeout(long time)
{
    // send the infoReady callback to the client
    callback.infoReady(m_sName);
}
}
```

SlowServiceControl.jcx Sample

This topic includes the source code for the SlowServiceControl.jcx Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/controlFactory/

Sample Source Code

```
package controlFactory;

/**
 * <p>A dummy service that represents a back end service that takes an psuedo-random amount of
 * @jc:location http-url="SlowService.jws" jms-url="SlowService.jws"
 * @jc:wSDL file="#SlowServiceWSDL"
 * @editor-info:link autogen-style="java" source="SlowService.jws" autogen="true"
 */
public interface SlowServiceControl extends com.bea.control.ControlExtension, com.bea.control.S
{
    public static class StartHeader
        implements java.io.Serializable
    {
        public java.lang.String conversationID;
        public java.lang.String callbackLocation;
    }

    public static class ContinueHeader
        implements java.io.Serializable
    {
        public java.lang.String conversationID;
    }

    public static class CallbackHeader
        implements java.io.Serializable
    {
        public java.lang.String conversationID;
    }

    public interface Callback extends com.bea.control.ServiceControl.Callback
    {
        /**
         * Callback that is sent to the client when this service is done doing its work.
         * @jc:conversation phase="finish"
         */
        public void infoReady (java.lang.String name);
    }

    /**
     * <p>The request from the client to do some work. After some amount of time, the infoReady
     * @jc:conversation phase="start"
     */
    public long requestInfo (java.lang.String name);
}
```

navWebServices.html Sample

```
static final long serialVersionUID = 1L;
}

/** @common:define name="SlowServiceWsd1" value::
<?xml version="1.0" encoding="utf-8"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:conv="http://www.openuri.org/2002/02/01/conv"
    <types>
        <s:schema elementFormDefault="qualified" targetNamespace="http://workshop.bea.com/SlowServiceWsd1">
            <s:element name="infoReadyResponse">
                <s:complexType>
                    <s:sequence/>
                </s:complexType>
            </s:element>
            <s:element name="infoReady">
                <s:complexType>
                    <s:sequence>
                        <s:element name="name" type="s:string" minOccurs="0"/>
                    </s:sequence>
                </s:complexType>
            </s:element>
            <s:element name="requestInfo">
                <s:complexType>
                    <s:sequence>
                        <s:element name="name" type="s:string" minOccurs="0"/>
                    </s:sequence>
                </s:complexType>
            </s:element>
            <s:element name="requestInfoResponse">
                <s:complexType>
                    <s:sequence>
                        <s:element name="requestInfoResult" type="s:long"/>
                    </s:sequence>
                </s:complexType>
            </s:element>
            <s:element name="long" type="s:long"/>
        </s:schema>

        <s:schema elementFormDefault="qualified" targetNamespace="http://www.openuri.org/2002/02/01/conv">
            <s:element name="StartHeader" type="conv:StartHeader"/>
            <s:element name="ContinueHeader" type="conv:ContinueHeader"/>
            <s:element name="CallbackHeader" type="conv:CallbackHeader"/>
            <s:complexType name="StartHeader">
                <s:sequence>
                    <s:element minOccurs="0" maxOccurs="1" name="conversationID" type="s:string"/>
                    <s:element minOccurs="0" maxOccurs="1" name="callbackLocation" type="s:string"/>
                </s:sequence>
            </s:complexType>
            <s:complexType name="ContinueHeader">
                <s:sequence>
                    <s:element minOccurs="1" maxOccurs="1" name="conversationID" type="s:string"/>
                </s:sequence>
            </s:complexType>
            <s:complexType name="CallbackHeader">
                <s:sequence>
                    <s:element minOccurs="1" maxOccurs="1" name="conversationID" type="s:string"/>
                </s:sequence>
            </s:complexType>
        </s:schema>
    </types>
    <message name="infoReadySoapIn">
```


navWebServices.html Sample

```
<part name="parameters" element="s0:infoReadyResponse"/>
</message>
<message name="infoReadySoapOut">
  <part name="parameters" element="s0:infoReady"/>
</message>
<message name="requestInfoSoapIn">
  <part name="parameters" element="s0:requestInfo"/>
</message>
<message name="requestInfoSoapOut">
  <part name="parameters" element="s0:requestInfoResponse"/>
</message>
<message name="infoReadyHttpGetIn"/>
<message name="infoReadyHttpGetOut">
  <part name="name" type="s:string"/>
</message>
<message name="requestInfoHttpGetIn">
  <part name="name" type="s:string"/>
</message>
<message name="requestInfoHttpGetOut">
  <part name="Body" element="s0:long"/>
</message>
<message name="infoReadyHttpPostIn"/>
<message name="infoReadyHttpPostOut">
  <part name="name" type="s:string"/>
</message>
<message name="requestInfoHttpPostIn">
  <part name="name" type="s:string"/>
</message>
<message name="requestInfoHttpPostOut">
  <part name="Body" element="s0:long"/>
</message>
<message name="StartHeader_literal">
  <part name="StartHeader" element="conv:StartHeader"/>
</message>
<message name="CallbackHeader_literal">
  <part name="CallbackHeader" element="conv:CallbackHeader"/>
</message>
<portType name="SlowServiceSoap">
  <operation name="infoReady">
    <documentation>Callback that is sent to the client when this service is done doing it</documentation>
    <output message="s0:infoReadySoapOut"/>
    <input message="s0:infoReadySoapIn"/>
  </operation>
  <operation name="requestInfo">
    <documentation><lt;p>&gt;The request from the client to do some work. After some amount of time, the client will receive a response.</p></documentation>
    <input message="s0:requestInfoSoapIn"/>
    <output message="s0:requestInfoSoapOut"/>
  </operation>
</portType>
<portType name="SlowServiceHttpGet">
  <operation name="infoReady">
    <documentation>Callback that is sent to the client when this service is done doing it</documentation>
    <output message="s0:infoReadyHttpGetOut"/>
    <input message="s0:infoReadyHttpGetIn"/>
  </operation>
  <operation name="requestInfo">
    <documentation><lt;p>&gt;The request from the client to do some work. After some amount of time, the client will receive a response.</p></documentation>
    <input message="s0:requestInfoHttpGetIn"/>
    <output message="s0:requestInfoHttpGetOut"/>
  </operation>
</portType>
```

navWebServices.html Sample

```

<portType name="SlowServiceHttpPost">
  <operation name="infoReady">
    <documentation>Callback that is sent to the client when this service is done doing it
    <output message="s0:infoReadyHttpPostOut"/>
    <input message="s0:infoReadyHttpPostIn"/>
  </operation>
  <operation name="requestInfo">
    <documentation><lt;p>&gt;The request from the client to do some work. After some amount of time, the
    <input message="s0:requestInfoHttpPostIn"/>
    <output message="s0:requestInfoHttpPostOut"/>
  </operation>
</portType>
<binding name="SlowServiceSoap" type="s0:SlowServiceSoap">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
  <operation name="infoReady">
    <soap:operation soapAction="http://workshop.bea.com/SlowService/infoReady" style="document"/>
    <cw:transition phase="finish"/>
    <input>
      <soap:body use="literal"/>
    </input>
    <output>
      <soap:body use="literal"/>
      <soap:header wsdl:required="true" message="s0:CallbackHeader_literal" part="CallbackHeader"/>
    </output>
  </operation>
  <operation name="requestInfo">
    <soap:operation soapAction="http://workshop.bea.com/SlowService/requestInfo" style="document"/>
    <cw:transition phase="start"/>
    <input>
      <soap:body use="literal"/>
      <soap:header wsdl:required="true" message="s0:StartHeader_literal" part="StartHeader"/>
    </input>
    <output>
      <soap:body use="literal"/>
    </output>
  </operation>
</binding>
<binding name="SlowServiceHttpGet" type="s0:SlowServiceHttpGet">
  <http:binding verb="GET"/>
  <operation name="infoReady">
    <http:operation location="/infoReady"/>
    <cw:transition phase="finish"/>
    <input>
      <mime:mimeType part="Body"/>
    </input>
    <output>
      <http:contentType"/>
    </output>
  </operation>
  <operation name="requestInfo">
    <http:operation location="/requestInfo"/>
    <cw:transition phase="start"/>
    <input>
      <http:contentType"/>
    </input>
    <output>
      <mime:mimeType part="Body"/>
    </output>
  </operation>
</binding>
<binding name="SlowServiceHttpPost" type="s0:SlowServiceHttpPost">

```

navWebServices.html Sample

```
<http:binding verb="POST"/>
<operation name="infoReady">
  <http:operation location="/infoReady"/>
  <cw:transition phase="finish"/>
  <input>
    <mime:mimeType part="Body"/>
  </input>
  <output>
    <mime:contentType type="application/x-www-form-urlencoded"/>
  </output>
</operation>
<operation name="requestInfo">
  <http:operation location="/requestInfo"/>
  <cw:transition phase="start"/>
  <input>
    <mime:contentType type="application/x-www-form-urlencoded"/>
  </input>
  <output>
    <mime:mimeType part="Body"/>
  </output>
</operation>
</binding>
<service name="SlowService">
  <documentation><p>A dummy service that represents a back end service that takes a
  <port name="SlowServiceSoap" binding="s0:SlowServiceSoap">
    <soap:address location="http://localhost:7001/controlFactory/SlowService.jws"/>
  </port>
  <port name="SlowServiceHttpGet" binding="s0:SlowServiceHttpGet">
    <http:address location="http://localhost:7001/controlFactory/SlowService.jws"/>
  </port>
  <port name="SlowServiceHttpPost" binding="s0:SlowServiceHttpPost">
    <http:address location="http://localhost:7001/controlFactory/SlowService.jws"/>
  </port>
</service>
</definitions>
* ::
*/
```

Controller.jspf Sample

This topic includes the source code for the Controller.jspf Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/

Sample Source Code

```
import com.bea.wlw.netui.pageflow.PageFlowController;
import com.bea.wlw.netui.pageflow.Forward;

/**
 * This is the default controller for a blank web application.
 * @jpf:view-properties view-properties::
 * <!-- This data is auto-generated. Hand-editing this section is not recommended. -->
 * <view-properties>
 * <pageflow-object id="pageflow:/Controller.jspf"/>
 * <pageflow-object id="action:begin.do">
 *   <property value="60" name="x"/>
 *   <property value="80" name="y"/>
 * </pageflow-object>
 * <pageflow-object id="page:index.jsp">
 *   <property value="240" name="x"/>
 *   <property value="80" name="y"/>
 * </pageflow-object>
 * <pageflow-object id="page:error.jsp">
 *   <property value="240" name="x"/>
 *   <property value="180" name="y"/>
 * </pageflow-object>
 * <pageflow-object id="forward:path#index#index.jsp#@action:begin.do@">
 *   <property value="index" name="label"/>
 *   <property value="West_1" name="toPort"/>
 *   <property value="East_1" name="fromPort"/>
 *   <property value="96,150,150,204" name="elbowsX"/>
 *   <property value="72,72,72,72" name="elbowsY"/>
 * </pageflow-object>
 * </view-properties>
 * ::
 *
 */
public class Controller extends PageFlowController
{
    /**
     * @jpf:action
     * @jpf:forward name="index" path="index.jsp"
     */
    protected Forward begin()
    {
        return new Forward("index");
    }
}
```

controlProjectTest Samples

This section contains source code for the following samples.

Samples Included in This Section

ControlTest.jws Sample

ControlTest.jws Sample

This topic includes the source code for the ControlTest.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/controlProjectTest/

Sample Source Code

```
package controlProjectTest;

/**
 * A web service to test the controls built from the JavaControlProject
 * project in the SamplesApp application. <br/><br/>
 *
 * If you change code in those controls to try other scenarios,
 * remember to build JavaControlProject (right-click JavaControlProject,
 * then click Build JavaControlProject). When you build that project, the
 * resulting JavaControlProject.jar file is copied to the Libraries of
 * the SamplesApp application. From there, they are accessible to
 * this web service. <br/><br/>
 *
 * Note that the WebServices project also includes local versions
 * of these controls, along with a corresponding test web service.
 * @common:target-namespace namespace="http://workshop.bea.com/ControlTest"
 */
public class ControlTest implements com.bea.jws.WebService
{
    public Callback callback;

    /**
     * @common:control
     */
    private verifyFunds.VerifyFunds verifyFundsControl;

    /**
     * Tests the VerifyFunds control, which submits purchase
     * order information after ensuring that inventory and
     * the customer's account support it. <br/><br/>
     *
     * To test the control, enter a PO number, customer ID,
     * item number, and item quantity. After clicking testVerifyFunds,
     * click Refresh until Test View displays the results of the
     * test as a message returned by the verifyFundsTestDone callback.
     *
     * The following table lists item numbers and quantities known
     * to the database when WebLogic Workshop is installed. The
     * VerifyFunds control decreases quantities with successful
     * orders. This web service gives the customer a balance
     * of $200.00 to spend for each request. You can change
     * this amount to test with another.
     *
     * <table style="font-size: 8 pt">
     * <tr><td style="width: 100px">itemNumber</td><td style="width: 100px">quantity</td><td>pr
```

navWebServices.html Sample

```
*
* <tr><td>624</td><td>5</td><td>34.95</td></tr>
* <tr><td>625</td><td>6</td><td>39.95</td></tr>
* <tr><td>629</td><td>10</td><td>34.95</td></tr>
* <tr><td>631</td><td>15</td><td>65.95</td></tr>
* <tr><td>640</td><td>1</td><td>24.95</td></tr>
*
* </table>
*
* @param poNumber A number for this purchase order. May be any
* number, and may include non-numeric characters (these will be removed
* by the control).
* @param customerID The customer's unique ID. This may be any number.
* @param item The number for the item being ordered. Use a number from
* the range given above.
* @param quantity The number of the specified items to order. Note that
* this number must be lower than the quantities given above in order
* for this order to succeed. Give larger numbers to test the control's
* inventory checking. Also, keep in mind that a successful PO transaction
* will decrease the inventory quantity in the database by the amount
* ordered.
* @common:operation
* @jws:conversation phase="start"
* @common:message-buffer enable="true"
*/
public void testVerifyFunds(String poNumber,
    String customerID, int item, int quantity)
{
    /*
    * The amount the customer can spend. Changing this amount may
    * change the results of the control's check to verify customer
    * funds for the purchase.
    */
    double balance = 200.00;

    /* Call the VerifyFunds control with the details of the purchase order. */
    verifyFundsControl.submitPO(poNumber, customerID, item, quantity, balance);
}

/*
* A handler for the VerifyFunds control's onTransactionComplete callback.
* The results are essentially summarized for a callback of this web service.
*/
public void verifyFundsControl_onTransactionComplete(java.lang.String message,
    boolean balanceAvailable, boolean inventoryAvailable)
{
    callback.verifyFundsTestDone(message +
        "\n Balance available: " + balanceAvailable +
        "\n Inventory available: " + inventoryAvailable);
}

public interface Callback
{
    /**
    * Sends a response to the client of this web service when the VerifyFunds
    * control's callback is received.
    *
    * @jws:conversation phase="finish"
    */
    void verifyFundsTestDone(String responseMessage);
}
```

```
}  
}
```


creditReport Samples

This section contains source code for the following samples.

Samples Included in This Section

Bank.jws Sample

BankControl.jcx Sample

CreditReport.jws Sample

IRS.jws Sample

IRSControl.jcx Sample

Bank.jws Sample

This topic includes the source code for the Bank.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/creditReport/

Sample Source Code

```
package creditReport;

/**
 * Bank.jws is a very simple service for use by the CreditReport.jws sample
 * service. Bank simulates a "long-running" procedure by using a timer
 * to delay its asynchronous response.
 *
 * Note that Bank.jws and IRS.jws are identical except for method names
 * and the default duration of the timer timeout.
 * @common:target-namespace namespace="http://workshop.bea.com/Bank"
 */
public class Bank implements com.bea.jws.WebService
{
    /**
     * @jc:timer timeout="10 seconds"
     * @common:control
     */
    private com.bea.control.TimerControl timer;

    /**
     * Store the customer ID number that is passed in so we can use it in the
     * asynchronous response.
     */
    private String ssn;

    public Callback callback;

    public interface Callback
    {
        /**
         * <p>onDeliverAnalysis is a callback delivered to the client when
         * processing is complete.</p>
         *
         * @jws:conversation phase="finish"
         */
        public void onDeliverAnalysis(String result);
    }

    /**
     * <p>Starts the asynchronous analysis operation. When analysis is complete
     * the onDeliverAnalysis callback will be called. If cancelAnalysis is
     * invoked before the results are delivered, results will never be delivered.</p>
     *
     * @common:operation
     * @jws:conversation phase="start"
     */
}
```

navWebServices.html Sample

```
*/
public void startCustomerAnalysis(String ssn)
{
    /* store the customer ID for later use */
    this.ssn = ssn;

    /*
     * start the timer, which simulates starting the long-running analysis procedure
     */
    timer.start();
}

/*
 * Handler for onTimeout events from the "timer" Timer Control.
 * onTimeout simulates the long-running procedure reaching completion,
 * so a result is sent to the client via the onDeliverAnalysis callback.
 */
private void timer_onTimeout(long timeout)
{
    /*
     * Use a completely arbitrary scheme to decide who's approved and
     * who's not. Some people think real banks work this way.
     */
    if ((ssn.length() > 2) && (ssn.charAt(2) <= '5'))
    {
        callback.onDeliverAnalysis("Approved");
    }
    else
    {
        callback.onDeliverAnalysis("Denied");
    }
}

/**
 * <p>Cancels the analysis. The onDeliverAnalysis callback
 * will not be called and the conversation is finished.</p>
 *
 * @common:operation
 * @jws:conversation phase="finish"
 */
public void cancelAnalysis()
{
    timer.stop();
}

}
```

BankControl.jcx Sample

This topic includes the source code for the BankControl.jcx Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/creditReport/

Sample Source Code

```
package creditReport;

/**
 * Bank.jws is a very simple service for use by the CreditReport.jws sample service. Bank simul
 * @jc:location http-url="Bank.jws" jms-url="Bank.jws"
 * @jc:wSDL file="#BankWSDL"
 * @editor-info:link autogen-style="java" source="Bank.jws" autogen="true"
 */
public interface BankControl extends com.bea.control.ControlExtension, com.bea.control.ServiceC
{
    public static class StartHeader
        implements java.io.Serializable
    {
        public java.lang.String conversationID;
        public java.lang.String callbackLocation;
    }

    public static class ContinueHeader
        implements java.io.Serializable
    {
        public java.lang.String conversationID;
    }

    public static class CallbackHeader
        implements java.io.Serializable
    {
        public java.lang.String conversationID;
    }

    public interface Callback extends com.bea.control.ServiceControl.Callback
    {
        /**
         * <p>onDeliverAnalysis is a callback delivered to the client when processing is comple
         * @jc:conversation phase="finish"
         */
        public void onDeliverAnalysis (java.lang.String result);
    }

    /**
     * <p>Starts the asynchronous analysis operation. When analysis is complete the onDeliverAn
     * @jc:conversation phase="start"
     */
    public void startCustomerAnalysis (java.lang.String ssn);
}
```

navWebServices.html Sample

```

/**
 * <p>Cancels the analysis. The onDeliverAnalysis callback will not be called and the conver
 * @jc:conversation phase="finish"
 */
public void cancelAnalysis ();

static final long serialVersionUID = 1L;
}

/** @common:define name="BankWsd1" value::
<?xml version="1.0" encoding="utf-8"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:conv="http://www.openuri.org/20
<types>
  <s:schema elementFormDefault="qualified" targetNamespace="http://workshop.bea.com/Bank"
    <s:element name="onDeliverAnalysisResponse">
      <s:complexType>
        <s:sequence/>
      </s:complexType>
    </s:element>
    <s:element name="onDeliverAnalysis">
      <s:complexType>
        <s:sequence>
          <s:element name="result" type="s:string" minOccurs="0"/>
        </s:sequence>
      </s:complexType>
    </s:element>
    <s:element name="startCustomerAnalysis">
      <s:complexType>
        <s:sequence>
          <s:element name="ssn" type="s:string" minOccurs="0"/>
        </s:sequence>
      </s:complexType>
    </s:element>
    <s:element name="startCustomerAnalysisResponse">
      <s:complexType>
        <s:sequence/>
      </s:complexType>
    </s:element>
    <s:element name="cancelAnalysis">
      <s:complexType>
        <s:sequence/>
      </s:complexType>
    </s:element>
    <s:element name="cancelAnalysisResponse">
      <s:complexType>
        <s:sequence/>
      </s:complexType>
    </s:element>
  </s:schema>

  <s:schema elementFormDefault="qualified" targetNamespace="http://www.openuri.org/2002/0
    <s:element name="StartHeader" type="conv:StartHeader"/>
    <s:element name="ContinueHeader" type="conv:ContinueHeader"/>
    <s:element name="CallbackHeader" type="conv:CallbackHeader"/>
    <s:complexType name="StartHeader">
      <s:sequence>
        <s:element minOccurs="0" maxOccurs="1" name="conversationID" type="s:string"/>
        <s:element minOccurs="0" maxOccurs="1" name="callbackLocation" type="s:string"/>
      </s:sequence>
    </s:complexType>

```

navWebServices.html Sample

```
<s:complexType name="ContinueHeader">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="conversationID" type="s:string"/>
  </s:sequence>
</s:complexType>
<s:complexType name="CallbackHeader">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="conversationID" type="s:string"/>
  </s:sequence>
</s:complexType>
</s:schema>
</types>
<message name="onDeliverAnalysisSoapIn">
  <part name="parameters" element="s0:onDeliverAnalysisResponse"/>
</message>
<message name="onDeliverAnalysisSoapOut">
  <part name="parameters" element="s0:onDeliverAnalysis"/>
</message>
<message name="startCustomerAnalysisSoapIn">
  <part name="parameters" element="s0:startCustomerAnalysis"/>
</message>
<message name="startCustomerAnalysisSoapOut">
  <part name="parameters" element="s0:startCustomerAnalysisResponse"/>
</message>
<message name="cancelAnalysisSoapIn">
  <part name="parameters" element="s0:cancelAnalysis"/>
</message>
<message name="cancelAnalysisSoapOut">
  <part name="parameters" element="s0:cancelAnalysisResponse"/>
</message>
<message name="onDeliverAnalysisHttpGetIn"/>
<message name="onDeliverAnalysisHttpGetOut">
  <part name="result" type="s:string"/>
</message>
<message name="startCustomerAnalysisHttpGetIn">
  <part name="ssn" type="s:string"/>
</message>
<message name="startCustomerAnalysisHttpGetOut"/>
<message name="cancelAnalysisHttpGetIn"/>
<message name="cancelAnalysisHttpGetOut"/>
<message name="onDeliverAnalysisHttpPostIn"/>
<message name="onDeliverAnalysisHttpPostOut">
  <part name="result" type="s:string"/>
</message>
<message name="startCustomerAnalysisHttpPostIn">
  <part name="ssn" type="s:string"/>
</message>
<message name="startCustomerAnalysisHttpPostOut"/>
<message name="cancelAnalysisHttpPostIn"/>
<message name="cancelAnalysisHttpPostOut"/>
<message name="StartHeader_literal">
  <part name="StartHeader" element="conv:StartHeader"/>
</message>
<message name="ContinueHeader_literal">
  <part name="ContinueHeader" element="conv:ContinueHeader"/>
</message>
<message name="CallbackHeader_literal">
  <part name="CallbackHeader" element="conv:CallbackHeader"/>
</message>
<portType name="BankSoap">
  <operation name="onDeliverAnalysis">
```

navWebServices.html Sample

```

    <documentation>&lt;p&gt;onDeliverAnalysis is a callback delivered to the client when
    <output message="s0:onDeliverAnalysisSoapOut"/>
    <input message="s0:onDeliverAnalysisSoapIn"/>
</operation>
<operation name="startCustomerAnalysis">
    <documentation>&lt;p&gt;Starts the asynchronous analysis operation. When analysis is
    <input message="s0:startCustomerAnalysisSoapIn"/>
    <output message="s0:startCustomerAnalysisSoapOut"/>
</operation>
<operation name="cancelAnalysis">
    <documentation>&lt;p&gt;Cancels the analysis. The onDeliverAnalysis callback will not
    <input message="s0:cancelAnalysisSoapIn"/>
    <output message="s0:cancelAnalysisSoapOut"/>
</operation>
</portType>
<portType name="BankHttpGet">
    <operation name="onDeliverAnalysis">
        <documentation>&lt;p&gt;onDeliverAnalysis is a callback delivered to the client when
        <output message="s0:onDeliverAnalysisHttpGetOut"/>
        <input message="s0:onDeliverAnalysisHttpGetIn"/>
    </operation>
    <operation name="startCustomerAnalysis">
        <documentation>&lt;p&gt;Starts the asynchronous analysis operation. When analysis is
        <input message="s0:startCustomerAnalysisHttpGetIn"/>
        <output message="s0:startCustomerAnalysisHttpGetOut"/>
    </operation>
    <operation name="cancelAnalysis">
        <documentation>&lt;p&gt;Cancels the analysis. The onDeliverAnalysis callback will not
        <input message="s0:cancelAnalysisHttpGetIn"/>
        <output message="s0:cancelAnalysisHttpGetOut"/>
    </operation>
</portType>
<portType name="BankHttpPost">
    <operation name="onDeliverAnalysis">
        <documentation>&lt;p&gt;onDeliverAnalysis is a callback delivered to the client when
        <output message="s0:onDeliverAnalysisHttpPostOut"/>
        <input message="s0:onDeliverAnalysisHttpPostIn"/>
    </operation>
    <operation name="startCustomerAnalysis">
        <documentation>&lt;p&gt;Starts the asynchronous analysis operation. When analysis is
        <input message="s0:startCustomerAnalysisHttpPostIn"/>
        <output message="s0:startCustomerAnalysisHttpPostOut"/>
    </operation>
    <operation name="cancelAnalysis">
        <documentation>&lt;p&gt;Cancels the analysis. The onDeliverAnalysis callback will not
        <input message="s0:cancelAnalysisHttpPostIn"/>
        <output message="s0:cancelAnalysisHttpPostOut"/>
    </operation>
</portType>
<binding name="BankSoap" type="s0:BankSoap">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
    <operation name="onDeliverAnalysis">
        <soap:operation soapAction="http://workshop.bea.com/Bank/onDeliverAnalysis" style="document"
        <cw:transition phase="finish"/>
        <input>
            <soap:body use="literal"/>
        </input>
        <output>
            <soap:body use="literal"/>
            <soap:header wsdl:required="true" message="s0:CallbackHeader_literal" part="CallbackHeader"/>
        </output>
    </operation>
    <operation name="startCustomerAnalysis">
        <soap:operation soapAction="http://workshop.bea.com/Bank/startCustomerAnalysis" style="document"
        <cw:transition phase="finish"/>
        <input>
            <soap:body use="literal"/>
        </input>
        <output>
            <soap:body use="literal"/>
            <soap:header wsdl:required="true" message="s0:CallbackHeader_literal" part="CallbackHeader"/>
        </output>
    </operation>
    <operation name="cancelAnalysis">
        <soap:operation soapAction="http://workshop.bea.com/Bank/cancelAnalysis" style="document"
        <cw:transition phase="finish"/>
        <input>
            <soap:body use="literal"/>
        </input>
        <output>
            <soap:body use="literal"/>
            <soap:header wsdl:required="true" message="s0:CallbackHeader_literal" part="CallbackHeader"/>
        </output>
    </operation>
</binding>
</service>

```

navWebServices.html Sample

```
</operation>
<operation name="startCustomerAnalysis">
  <soap:operation soapAction="http://workshop.bea.com/Bank/startCustomerAnalysis" style="document" />
  <cw:transition phase="start" />
  <input>
    <soap:body use="literal" />
    <soap:header wsdl:required="true" message="s0:StartHeader_literal" part="StartHeader" />
  </input>
  <output>
    <soap:body use="literal" />
  </output>
</operation>
<operation name="cancelAnalysis">
  <soap:operation soapAction="http://workshop.bea.com/Bank/cancelAnalysis" style="document" />
  <cw:transition phase="finish" />
  <input>
    <soap:body use="literal" />
    <soap:header wsdl:required="true" message="s0:ContinueHeader_literal" part="ContinueHeader" />
  </input>
  <output>
    <soap:body use="literal" />
  </output>
</operation>
</binding>
<binding name="BankHttpGet" type="s0:BankHttpGet">
  <http:binding verb="GET" />
  <operation name="onDeliverAnalysis">
    <http:operation location="/onDeliverAnalysis" />
    <cw:transition phase="finish" />
    <input>
      <mime:mimeXml part="Body" />
    </input>
    <output>
      <http:urlEncoded />
    </output>
  </operation>
  <operation name="startCustomerAnalysis">
    <http:operation location="/startCustomerAnalysis" />
    <cw:transition phase="start" />
    <input>
      <http:urlEncoded />
    </input>
    <output />
  </operation>
  <operation name="cancelAnalysis">
    <http:operation location="/cancelAnalysis" />
    <cw:transition phase="finish" />
    <input>
      <http:urlEncoded />
    </input>
    <output />
  </operation>
</binding>
<binding name="BankHttpPost" type="s0:BankHttpPost">
  <http:binding verb="POST" />
  <operation name="onDeliverAnalysis">
    <http:operation location="/onDeliverAnalysis" />
    <cw:transition phase="finish" />
    <input>
      <mime:mimeXml part="Body" />
    </input>
  </operation>
</binding>
```


navWebServices.html Sample

```
<output>
  <mime:content type="application/x-www-form-urlencoded"/>
</output>
</operation>
<operation name="startCustomerAnalysis">
  <http:operation location="/startCustomerAnalysis"/>
  <cw:transition phase="start"/>
  <input>
    <mime:content type="application/x-www-form-urlencoded"/>
  </input>
  <output/>
</operation>
<operation name="cancelAnalysis">
  <http:operation location="/cancelAnalysis"/>
  <cw:transition phase="finish"/>
  <input>
    <mime:content type="application/x-www-form-urlencoded"/>
  </input>
  <output/>
</operation>
</binding>
<service name="Bank">
  <documentation>Bank.jws is a very simple service for use by the CreditReport.jws sample
  <port name="BankSoap" binding="s0:BankSoap">
    <soap:address location="http://localhost:7001/creditReport/Bank.jws"/>
  </port>
  <port name="BankHttpGet" binding="s0:BankHttpGet">
    <http:address location="http://localhost:7001/creditReport/Bank.jws"/>
  </port>
  <port name="BankHttpPost" binding="s0:BankHttpPost">
    <http:address location="http://localhost:7001/creditReport/Bank.jws"/>
  </port>
</service>
</definitions>
* ::
*/
```

CreditReport.jws Sample

This topic includes the source code for the CreditReport.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/creditReport/

Sample Source Code

```
package creditReport;

/**
 * <p>Credit reporting service. This service simulates constructing
 * a credit report from multiple secondary sources of information.
 * It uses two external services, one representing a bank and the
 * other representing the Internal Revenue Service (IRS).</p>
 *
 * <p>The @jws:conversation-lifetime max-idle-time tag controls how
 * long a conversational instance of this service will survive without
 * seeing activity.</p>
 *
 * <p>Conversations represent resources: they shouldn't be left around.</p>
 * @common:target-namespace namespace="http://workshop.bea.com/CreditReport"
 * @jws:conversation-lifetime max-age="1 hour" max-idle-time="30 minutes"
 * @common:xmlns namespace="http://openuri.org/bea/samples/workshop/creditReport" prefix="ns0"
 */

public class CreditReport implements com.bea.jws.WebService
{
    /**
     * @common:control
     */
    private creditReport.IRSControl irsControl;

    /**
     * @common:control
     */
    private creditReport.BankControl bankControl;

    /**
     * <p>ReportResult is an inner class used to represent a complete
     * credit report. It is used to return report responses to
     * the client.</p>
     *
     * <p>Inner classes that are used outside the parent class
     * MUST be declared "public static" and they MUST have
     * a public no-argument constructor.</p>
     */
    public static class ReportResult implements java.io.Serializable
    {
        public String bank;
        public String tax;
    }
}
```

navWebServices.html Sample

```
    public ReportResult() {}
}

/**
 * <p>Here we declare some persistent storage used
 * to store intermediate results.</p>
 *
 * <p>All member variables of the Java class become
 * persistent in the presence of a @common:conversation
 * start tag on one or more methods.</p>
 */
public String taxReport = null;
public String bankReport = null;

public Callback callback;

/**
 * <p>A Service can have an inner interface called
 * Callback if it wants to have automatic (and WSDL)
 * support for sending asynchronous messages back to
 * the client.</p>
 *
 * <p>The methods in the callback interface are published in
 * the .wsdl file for this service.</p>
 *
 * <p>Note that callbacks are asynchronous messages to the
 * client. It may not be possible for these messages
 * to pass through firewalls.</p>
 */
public interface Callback
{
    /**
     * <p>We call this method whenever we get some new data, to
     * tell our client what kind of new data we have. We
     * don't transmit the complete data, just a status
     * message.</p>
     *
     * <p>We've declared this as a buffered method so that
     * it's queued when we send it. That way we don't sit
     * around waiting for the client to process it. Only
     * methods that return 'void' may be buffered.</p>
     *
     * @jws:conversation phase="continue"
     * @common:message-buffer enable="true"
     */
    void onProgressNotify(String progressMsg);

    /**
     * <p>We call this method to deliver the final credit
     * report once we're completely done (both external
     * services have completed).</p>
     *
     * <p>We define the format of the message this
     * callback contains by using a custom XQuery map.
     * An XQuery map is like an XML map, except that it
     * uses XQuery expressions to retrieve values that should
     * be inserted into the message. </p>
     */
}
```

navWebServices.html Sample

```
*
* @jws:conversation phase="finish"
*
* @common:message-buffer enable="true"
* @jws:parameter-xml schema-element="ns0:creditReport" xquery::
* declare namespace ns0="http://workshop.bea.com/CreditReport"
* declare namespace ns1="http://openuri.org/bea/samples/workshop/creditReport"

* <ns1:creditReport>
*   <ns1:bankReport>{data($input//ns0:bank)}</ns1:bankReport>
*   <ns1:taxReport>{data($input//ns0:tax)}</ns1:taxReport>
* </ns1:creditReport>
* ::
*/
void onReportDone(ReportResult resultMsg);
}

/**
* <p>Request a credit report from our service.</p>
*
* <p>Since it may take some time for us to complete
* the report (we have to wait until both of our
* requests to external services complete), this
* methods has no return value. We will return the
* result to the client at a later time via the
* onReportDone() callback, above.</p>
*
* <p>We start a conversation so that per-client
* information will be persisted and correlated.
* There may be several clients using this service
* simultaneously. Conversations keep track of
* data and message traffic on a per-client basis
* automatically.</p>
*
* <p>We add the @common:message-buffer tag so the client
* doesn't synchronously wait for a return from
* this request.</p>
*
* @common:operation
* @jws:conversation phase="start"
* @common:message-buffer enable="true"
*/
public void requestReport(String ssn)
{
    // Request information from the bank
    bankControl.startCustomerAnalysis(ssn);

    // Request information from the IRS
    irsControl.requestTaxReport(ssn);

    /*
    * Both of the external services will return
    * their responses to us via callbacks that
    * we will recieve via the handlers
    * bank_onDeliverAnalysis and irs_onDeliverTaxReport,
    * below. So we're done initiating the request.
    */
    return;
}

/**
```

navWebServices.html Sample

```
* <p>Request the current status of the request.</p>
*
* <p>For our impatient customers, we can give them
* partial results whenever they ask for them.</p>
*
* <p>Since this only makes sense after a request
* is initiated (ie., a conversation is started),
* this is a continue method that runs
* in the context of an existing conversation.</p>
*
* <p>The current result is delivered as an XML
* message with its format controlled by
* an XQuery map.</p>
*
* @common:operation
* @jws:conversation phase="continue"
* @jws:return-xml schema-element="ns0:creditReportReturn" xquery::
* declare namespace ns0="http://workshop.bea.com/CreditReport"
* declare namespace ns1="http://openuri.org/bea/samples/workshop/creditReport"

* <ns1:creditReportReturn>
*   <ns1:bankReport>{data($input//ns0:bank)}</ns1:bankReport>
*   <ns1:taxReport>{data($input//ns0:tax)}</ns1:taxReport>
* </ns1:creditReportReturn>
* ::
*/
public ReportResult getCurrentStatus()
{
    /*
    * Call an internal helper method to construct
    * an intermediate result.
    */
    return assembleResult();
}

/**
* <p>Cancel the credit report request.</p>
*
* <p>This is marked as a finish method so that the
* conversation is automatically ended when we
* finish processing this message.</p>
*
* <p>We've declared this as a buffered method so the
* client doesn't sit around waiting for a return.</p>
*
* @common:operation
* @jws:conversation phase="finish"
* @common:message-buffer enable="true"
*/
public void cancelReport()
{
    /*
    * If we haven't heard from the bank, cancel
    * the request to the bank.
    */
    if (bankReport == null)
    {
        bankControl.cancelAnalysis();
    }
}

/*
```

navWebServices.html Sample

```
* If we haven't heard from the IRS, cancel
* the request to the IRS.
*/
if (taxReport == null)
{
    irsControl.cancelReport();
}
}

/**
 * <p>Handler for the onDeliverAnalysis callback of the Bank.jws
 * web service.</p>
 *
 * <p>The name of the handler is always the variable name of the
 * service control ('bank' in this case), followed by an
 * underscore and the name of the callback.</p>
 */
public void bankControl_onDeliverAnalysis(String analysisResult)
{
    /*
     * Store the bank result in persistent storage.
     */
    bankReport = analysisResult;

    /*
     * Notify the client that we have recieved results
     * from the bank.
     */
    callback.onProgressNotify("Bank Report Received");

    /*
     * Check to see if all results have been received.
     * If so, checkIfDone will send the completed
     * results.
     */
    checkIfDone();
}

/**
 * <p>Handler for the onDeliverTaxReport callback of the IRS.jws
 * web service.</p>
 *
 * <p>The name of the handler is always the variable name of the
 * service control ('irs' in this case), followed by an
 * underscore and the name of the callback.</p>
 */
public void irsControl_onDeliverTaxReport(String report)
{
    /*
     * Store the IRS result in persistent storage.
     */
    taxReport = report;

    /*
     * Notify the client that we have recieved results
     * from the IRS.
     */
    callback.onProgressNotify("Tax Report Received");

    /*
     * Check to see if all results have been received.
```

navWebServices.html Sample

```
    * If so, checkIfDone will send the completed
    * results.
    */
    checkIfDone();
}

/**
 * A private helper that builds our simple credit report,
 * using the current (possibly incomplete) results.
 */
private ReportResult assembleResult()
{
    ReportResult result = new ReportResult();

    /*
     * Java includes the ?: construct as a shorthand way of
     * coding if-then-else. The statement takes the form:
     *
     * value = condition ? value-if-true : value-if-false
     *
     * If condition evaluates to true, the value-if-true
     * expression is evaluated and the result is assigned
     * to value. If condition evaluates to false, the
     * value-if-false expression is evaluated and the
     * result is assigned to value.
     */
    result.bank = (bankReport == null) ? "Not Received" : bankReport;
    result.tax = (taxReport == null) ? "Not Received" : taxReport;

    return result;
}

/**
 * A private helper that may call "callback.onReportDone" if
 * we're actually done.
 */
private void checkIfDone()
{
    /*
     * If all results are back from external services...
     */
    if (bankReport != null && taxReport != null)
    {
        /*
         * ... call the client with the completed results.
         */
        callback.onReportDone(assembleResult());
    }
    return;
}
}
```

IRS.jws Sample

This topic includes the source code for the IRS.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/creditReport/

Sample Source Code

```
package creditReport;

/**
 * IRS.jws is a very simple service for use by the CreditReport.jws sample
 * service.  IRS simulates a "long-running" procedure by using a timer
 * to delay its asynchronous response.
 *
 * Note that Bank.jws and IRS.jws are identical except for method names
 * and the default duration of the timer timeout.
 * @common:target-namespace namespace="http://workshop.bea.com/IRS"
 */

public class IRS implements com.bea.jws.WebService
{
    /**
     * <p>A Timer Control is used to simulate an operation that takes some time.</p>
     *
     * @jc:timer timeout="10 seconds"
     * @common:control
     */
    private com.bea.control.TimerControl timer;

    /**
     * Store the customer ID number that is passed in so we can use it in the
     * asynchronous response.
     */
    private String ssn;

    public Callback callback;

    public interface Callback
    {
        /**
         * onDeliverTaxReport is a callback delivered to the client when
         * processing is complete.
         */
        /**
         * @jws:conversation phase="finish"
         */
        public void onDeliverTaxReport(String result);
    }
}
```


navWebServices.html Sample

```
/**
 * <p>Starts the asynchronous tax report operation. When it is complete
 * the onDeliverTaxReport callback will be called. If cancelReport is
 * invoked before the results are delivered, results will never be delivered.</p>
 *
 * @common:operation
 * @jws:conversation phase="start"
 */
public void requestTaxReport(String ssn)
{
    /* store the customer ID for later use */
    this.ssn = ssn;

    /*
     * start the timer, which simulates starting the long-running procedure
     */
    timer.start();
}

/*
 * Handler for onTimeout events from the "timer" Timer Control.
 * onTimeout simulates the long-running procedure reaching completion,
 * so a result is sent to the client via the onDeliverAnalysis callback.
 */
private void timer_onTimeout(long timeout)
{
    /*
     * Use a completely arbitrary scheme to decide who's approved and
     * who's not. Some people think real banks work this way.
     */
    if ((ssn.length() > 4) && (ssn.charAt(3) <= '5'))
    {
        callback.onDeliverTaxReport("Taxes Delinquent");
    }
    else
    {
        callback.onDeliverTaxReport("Taxes Current");
    }
}

/**
 * <p>Cancels the request. The onDeliverTaxReport callback
 * will not be called and the conversation is finished.</p>
 *
 * @common:operation
 * @jws:conversation phase="finish"
 */
public void cancelReport()
{
    timer.stop();
}
}
```

IRSControl.jcx Sample

This topic includes the source code for the IRSControl.jcx Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/creditReport/

Sample Source Code

```
package creditReport;

/**
 * IRS.jws is a very simple service for use by the CreditReport.jws sample service. IRS simulat
 * @jc:location http-url="IRS.jws" jms-url="IRS.jws"
 * @jc:wSDL file="#IRSWSdl"
 * @editor-info:link autogen-style="java" source="IRS.jws" autogen="true"
 */
public interface IRSControl extends com.bea.control.ControlExtension, com.bea.control.ServiceCo
{
    public static class StartHeader
        implements java.io.Serializable
    {
        public java.lang.String conversationID;
        public java.lang.String callbackLocation;
    }

    public static class ContinueHeader
        implements java.io.Serializable
    {
        public java.lang.String conversationID;
    }

    public static class CallbackHeader
        implements java.io.Serializable
    {
        public java.lang.String conversationID;
    }

    public interface Callback extends com.bea.control.ServiceControl.Callback
    {
        /**
         * @jc:conversation phase="finish"
         */
        public void onDeliverTaxReport (java.lang.String result);
    }

    /**
     * <p>Starts the asynchronous tax report operation. When it is complete the onDeliverTaxRep
     * @jc:conversation phase="start"
     */
    public void requestTaxReport (java.lang.String ssn);
}
```

navWebServices.html Sample

```

/**
 * <p>Cancels the request. The onDeliverTaxReport callback will not be called and the conver
 * @jc:conversation phase="finish"
 */
public void cancelReport ();

static final long serialVersionUID = 1L;
}

/** @common:define name="IRSWsdl" value::
<?xml version="1.0" encoding="utf-8"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:conv="http://www.openuri.org/20
<types>
  <s:schema elementFormDefault="qualified" targetNamespace="http://workshop.bea.com/IRS"
    <s:element name="onDeliverTaxReportResponse">
      <s:complexType>
        <s:sequence/>
      </s:complexType>
    </s:element>
    <s:element name="onDeliverTaxReport">
      <s:complexType>
        <s:sequence>
          <s:element name="result" type="s:string" minOccurs="0"/>
        </s:sequence>
      </s:complexType>
    </s:element>
    <s:element name="requestTaxReport">
      <s:complexType>
        <s:sequence>
          <s:element name="ssn" type="s:string" minOccurs="0"/>
        </s:sequence>
      </s:complexType>
    </s:element>
    <s:element name="requestTaxReportResponse">
      <s:complexType>
        <s:sequence/>
      </s:complexType>
    </s:element>
    <s:element name="cancelReport">
      <s:complexType>
        <s:sequence/>
      </s:complexType>
    </s:element>
    <s:element name="cancelReportResponse">
      <s:complexType>
        <s:sequence/>
      </s:complexType>
    </s:element>
  </s:schema>

  <s:schema elementFormDefault="qualified" targetNamespace="http://www.openuri.org/2002/0
    <s:element name="StartHeader" type="conv:StartHeader"/>
    <s:element name="ContinueHeader" type="conv:ContinueHeader"/>
    <s:element name="CallbackHeader" type="conv:CallbackHeader"/>
    <s:complexType name="StartHeader">
      <s:sequence>
        <s:element minOccurs="0" maxOccurs="1" name="conversationID" type="s:string"/>
        <s:element minOccurs="0" maxOccurs="1" name="callbackLocation" type="s:string"/>
      </s:sequence>
    </s:complexType>
    <s:complexType name="ContinueHeader">

```

navWebServices.html Sample

```
<s:sequence>
  <s:element minOccurs="1" maxOccurs="1" name="conversationID" type="s:string"/>
</s:sequence>
</s:complexType>
<s:complexType name="CallbackHeader">
  <s:sequence>
    <s:element minOccurs="1" maxOccurs="1" name="conversationID" type="s:string"/>
  </s:sequence>
</s:complexType>
</s:schema>
</types>
<message name="onDeliverTaxReportSoapIn">
  <part name="parameters" element="s0:onDeliverTaxReportResponse"/>
</message>
<message name="onDeliverTaxReportSoapOut">
  <part name="parameters" element="s0:onDeliverTaxReport"/>
</message>
<message name="requestTaxReportSoapIn">
  <part name="parameters" element="s0:requestTaxReport"/>
</message>
<message name="requestTaxReportSoapOut">
  <part name="parameters" element="s0:requestTaxReportResponse"/>
</message>
<message name="cancelReportSoapIn">
  <part name="parameters" element="s0:cancelReport"/>
</message>
<message name="cancelReportSoapOut">
  <part name="parameters" element="s0:cancelReportResponse"/>
</message>
<message name="onDeliverTaxReportHttpGetIn"/>
<message name="onDeliverTaxReportHttpGetOut">
  <part name="result" type="s:string"/>
</message>
<message name="requestTaxReportHttpGetIn">
  <part name="ssn" type="s:string"/>
</message>
<message name="requestTaxReportHttpGetOut"/>
<message name="cancelReportHttpGetIn"/>
<message name="cancelReportHttpGetOut"/>
<message name="onDeliverTaxReportHttpPostIn"/>
<message name="onDeliverTaxReportHttpPostOut">
  <part name="result" type="s:string"/>
</message>
<message name="requestTaxReportHttpPostIn">
  <part name="ssn" type="s:string"/>
</message>
<message name="requestTaxReportHttpPostOut"/>
<message name="cancelReportHttpPostIn"/>
<message name="cancelReportHttpPostOut"/>
<message name="StartHeader_literal">
  <part name="StartHeader" element="conv:StartHeader"/>
</message>
<message name="ContinueHeader_literal">
  <part name="ContinueHeader" element="conv:ContinueHeader"/>
</message>
<message name="CallbackHeader_literal">
  <part name="CallbackHeader" element="conv:CallbackHeader"/>
</message>
<portType name="IRSSoap">
  <operation name="onDeliverTaxReport">
    <output message="s0:onDeliverTaxReportSoapOut"/>
  </operation>
</portType>
</binding>
</service>
```

navWebServices.html Sample

```

    <input message="s0:onDeliverTaxReportSoapIn"/>
  </operation>
  <operation name="requestTaxReport">
    <documentation><![CDATA[Starts the asynchronous tax report operation. When it is complete, the onDeliverTaxReport callback will be invoked.]]>
    <input message="s0:requestTaxReportSoapIn"/>
    <output message="s0:requestTaxReportSoapOut"/>
  </operation>
  <operation name="cancelReport">
    <documentation><![CDATA Cancels the request. The onDeliverTaxReport callback will not be invoked.]]>
    <input message="s0:cancelReportSoapIn"/>
    <output message="s0:cancelReportSoapOut"/>
  </operation>
</portType>
<portType name="IRSHttpGet">
  <operation name="onDeliverTaxReport">
    <output message="s0:onDeliverTaxReportHttpGetOut"/>
    <input message="s0:onDeliverTaxReportHttpGetIn"/>
  </operation>
  <operation name="requestTaxReport">
    <documentation><![CDATA[Starts the asynchronous tax report operation. When it is complete, the onDeliverTaxReport callback will be invoked.]]>
    <input message="s0:requestTaxReportHttpGetIn"/>
    <output message="s0:requestTaxReportHttpGetOut"/>
  </operation>
  <operation name="cancelReport">
    <documentation><![CDATA Cancels the request. The onDeliverTaxReport callback will not be invoked.]]>
    <input message="s0:cancelReportHttpGetIn"/>
    <output message="s0:cancelReportHttpGetOut"/>
  </operation>
</portType>
<portType name="IRSHttpPost">
  <operation name="onDeliverTaxReport">
    <output message="s0:onDeliverTaxReportHttpPostOut"/>
    <input message="s0:onDeliverTaxReportHttpPostIn"/>
  </operation>
  <operation name="requestTaxReport">
    <documentation><![CDATA[Starts the asynchronous tax report operation. When it is complete, the onDeliverTaxReport callback will be invoked.]]>
    <input message="s0:requestTaxReportHttpPostIn"/>
    <output message="s0:requestTaxReportHttpPostOut"/>
  </operation>
  <operation name="cancelReport">
    <documentation><![CDATA Cancels the request. The onDeliverTaxReport callback will not be invoked.]]>
    <input message="s0:cancelReportHttpPostIn"/>
    <output message="s0:cancelReportHttpPostOut"/>
  </operation>
</portType>
<binding name="IRSSoap" type="s0:IRSSoap">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
  <operation name="onDeliverTaxReport">
    <soap:operation soapAction="http://workshop.bea.com/IRS/onDeliverTaxReport" style="document"/>
    <cw:transition phase="finish"/>
    <input>
      <soap:body use="literal"/>
    </input>
    <output>
      <soap:body use="literal"/>
      <soap:header wsdl:required="true" message="s0:CallbackHeader_literal" part="CallbackHeader"/>
    </output>
  </operation>
  <operation name="requestTaxReport">
    <soap:operation soapAction="http://workshop.bea.com/IRS/requestTaxReport" style="document"/>
    <cw:transition phase="start"/>

```

navWebServices.html Sample

```

<input>
  <soap:body use="literal"/>
  <soap:header wsdl:required="true" message="s0:StartHeader_literal" part="StartHeader"/>
</input>
<output>
  <soap:body use="literal"/>
</output>
</operation>
<operation name="cancelReport">
  <soap:operation soapAction="http://workshop.bea.com/IRS/cancelReport" style="document" />
  <cw:transition phase="finish"/>
  <input>
    <soap:body use="literal"/>
    <soap:header wsdl:required="true" message="s0:ContinueHeader_literal" part="ContinueHeader"/>
  </input>
  <output>
    <soap:body use="literal"/>
  </output>
</operation>
</binding>
<binding name="IRSHttpGet" type="s0:IRSHttpGet">
  <http:binding verb="GET"/>
  <operation name="onDeliverTaxReport">
    <http:operation location="/onDeliverTaxReport"/>
    <cw:transition phase="finish"/>
    <input>
      <mime:mimeXml part="Body"/>
    </input>
    <output>
      <http:urlEncoded/>
    </output>
  </operation>
  <operation name="requestTaxReport">
    <http:operation location="/requestTaxReport"/>
    <cw:transition phase="start"/>
    <input>
      <http:urlEncoded/>
    </input>
    <output/>
  </operation>
  <operation name="cancelReport">
    <http:operation location="/cancelReport"/>
    <cw:transition phase="finish"/>
    <input>
      <http:urlEncoded/>
    </input>
    <output/>
  </operation>
</binding>
<binding name="IRSHttpPost" type="s0:IRSHttpPost">
  <http:binding verb="POST"/>
  <operation name="onDeliverTaxReport">
    <http:operation location="/onDeliverTaxReport"/>
    <cw:transition phase="finish"/>
    <input>
      <mime:mimeXml part="Body"/>
    </input>
    <output>
      <mime:content type="application/x-www-form-urlencoded"/>
    </output>
  </operation>

```

navWebServices.html Sample

```
<operation name="requestTaxReport">
  <http:operation location="/requestTaxReport"/>
  <cw:transition phase="start"/>
  <input>
    <mime:content type="application/x-www-form-urlencoded"/>
  </input>
  <output/>
</operation>
<operation name="cancelReport">
  <http:operation location="/cancelReport"/>
  <cw:transition phase="finish"/>
  <input>
    <mime:content type="application/x-www-form-urlencoded"/>
  </input>
  <output/>
</operation>
</binding>
<service name="IRS">
  <documentation>IRS.jws is a very simple service for use by the CreditReport.jws sample
  <port name="IRSSoap" binding="s0:IRSSoap">
    <soap:address location="http://localhost:7001/creditReport/IRS.jws"/>
  </port>
  <port name="IRSHttpGet" binding="s0:IRSHttpGet">
    <http:address location="http://localhost:7001/creditReport/IRS.jws"/>
  </port>
  <port name="IRSHttpPost" binding="s0:IRSHttpPost">
    <http:address location="http://localhost:7001/creditReport/IRS.jws"/>
  </port>
</service>
</definitions>
* ::
*/
```

database Samples

This section contains source code for the following samples.

Samples Included in This Section

customer_db Samples

dynamicSQL Samples

lucky_number_db Samples

xmlBean Samples

customer_db Samples

This section contains source code for the following samples.

Samples Included in This Section

Customer.java Sample

CustomerDB.jcx Sample

CustomerDBClient.jws Sample

Customer.java Sample

This topic includes the source code for the Customer.java Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/database/customer_db/

Sample Source Code

```
package database.customer_db;

/**
 * <p>This class reflects the structure of the Customer
 * table in the sample database. The table is created
 * by createCustomerTable(), below.</p>
 *
 * <p>This class may be used as a return value for
 * methods that return Customer records, but each
 * member name of this class must match a column
 * name in the table and be of compatible type.</p>
 */
public class Customer
{
    private int custid;
    private String name;
    private String address;
    private String city;
    private String state;
    private String zip;
    private String area_code;
    private String phone;

    public Customer() {}

    public Customer(int id, String sName, String sAddress, String sCity, String sState,
                    String sZip, String sAreaCode, String sPhone)
    {
        custid = id;
        name = new String(sName);
        address = new String(sAddress);
        city = new String(sCity);
        state = new String(sState);
        zip = new String(sZip);
        area_code = new String(sAreaCode);
        phone = new String(sPhone);
    }

    public int getCustid()
    {
        return this.custid;
    }

    public void setCustid(int custid)
    {

```

```
        this.custid = custid;
    }

    public String getName()
    {
        return this.name;
    }

    public void setName(String name)
    {
        this.name = name;
    }

    public String getAddress()
    {
        return this.address;
    }

    public void setAddress(String address)
    {
        this.address = address;
    }

    public String getCity()
    {
        return this.city;
    }

    public void setCity(String city)
    {
        this.city = city;
    }

    public String getState()
    {
        return this.state;
    }

    public void setState(String state)
    {
        this.state = state;
    }

    public String getZip()
    {
        return this.zip;
    }

    public void setZip(String zip)
    {
        this.zip = zip;
    }

    public String getArea_code()
    {
        return this.area_code;
    }

    public void setArea_code(String area_code)
    {
        this.area_code = area_code;
    }
}
```

navWebServices.html Sample

```
    }  
  
    public String getPhone()  
    {  
        return this.phone;  
    }  
  
    public void setPhone(String phone)  
    {  
        this.phone = phone;  
    }  
}
```

CustomerDB.jcx Sample

This topic includes the source code for the CustomerDB.jcx Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/database/customer_db/

Sample Source Code

```
package database.customer_db;

import com.bea.control.DatabaseControl;
import java.sql.SQLException;

/**
 * <p> A sample Database Control that demonstrates use of
 * CREATE TABLE, DROP TABLE, INSERT, UPDATE, and SELECT
 * SQL statements.</p>
 *
 * <p>This Database Control uses the JDBC Data Source
 * "cgSampleDataSource" which is configured automatically
 * when WebLogic Workshop is installed.</p>
 *
 * <p>The data source is configured with the JNDI
 * name "cgSampleDataSource".</p>
 *
 * @jc:connection data-source-jndi-name="cgSampleDataSource"
 */

public interface CustomerDB extends com.bea.control.ControlExtension, DatabaseControl
{

    /**
     * <p>Demonstrates SQL's 'CREATE TABLE' statement using a multi-line @jc:sql tag.</p>
     *
     * <p>Note that database column names match the member names in the Customer class
     * with the exception of case.</p>
     *
     * <p>The CUSTOMER table is created and populated when WebLogic Workshop is
     * installed. If this method is called when the table already exists, a
     * harmless SQLException will be thrown. This method is here to
     * illustrate that CREATE TABLE statements, like any SQL statement, can be
     * issued by a Database control method.</p>
     *
     * @return Creation or failure reason
     *
     * @jc:sql statement::
     * CREATE TABLE customer (
     *     custid INT
     *     CONSTRAINT customer_pk PRIMARY KEY,
     *     name VARCHAR(40),
     *     address VARCHAR(40),
     *     city VARCHAR(40),

```

navWebServices.html Sample

```
*         state CHAR(2),
*         zip VARCHAR(11),
*         area_code CHAR(3),
*         phone CHAR(9))
*         ::
*/
public void createCustomerTable() throws SQLException;

/**
 * <p>Demonstrates SQL DROP TABLE statement.</p>
 *
 * <p>The CUSTOMER table is created when WebLogic Workshop is installed and
 * should probably not be deleted. If this method is called when the table
 * does not exist, a harmless SQLException will be thrown. This
 * method is here to illustrate that DROP TABLE statements, like any SQL
 * statement, can be issued by a Database control method.</p>
 *
 * @return Creation or failure reason
 *
 * @jc:sql statement="DROP TABLE customer"
 */
public void dropCustomerTable() throws SQLException;

/**
 * <p>Demonstrates the throwing of a SQL exception.</p>
 *
 * <p>An intentionally malformed SQL statement is attempted.</p>
 *
 * @jc:sql statement="CREATE TABLE badTable missingParen INT"
 */
public void badSQL() throws java.sql.SQLException;

/**
 * <p>Demonstrates SQL INSERT statement using a multi-line @jc:sql tag.</p>
 *
 * @return The number of rows inserted.
 *
 * @param custid The key of the customer record.
 * @param name The new name for the customer record (40 characters).
 * @param address The new street address for the customer record (40 characters).
 * @param city The new city for the customer record (40 characters).
 * @param state The new 2 letter state code for the customer record.
 * @param zip The new 9 digit zip code (xxxxx-xxxx) for the customer record.
 * @param area_code The new 3 digit area code for the customer record.
 * @param phone The new 7 digit phone number (xxx-xxxx) for the customer record.
 *
 * @jc:sql statement::
 *     INSERT INTO customer (custid,name,address,city,state,zip,area_code,phone)
 *     VALUES ({custid},{name},{address},{city},{state},{zip},{area_code},{phone})
 *     ::
 */
public int insertCustomer(int custid, String name, String address, String city, String state, String phone)
throws SQLException;

/**
 * <p>Demonstrates SQL INSERT statement using a multi-line @jc:sql tag with
 * substitutions from an object.</p>
 *
 * @return The number of rows inserted.
 *
 * @param cust A Customer object whose member values will become the field values
 * of the record in the database.
```

navWebServices.html Sample

```
*
* @jc:sql statement::
*     INSERT INTO customer (custid,name,address,city,state,zip,area_code,phone)
*     VALUES ({cust.custid},{cust.name},{cust.address},{cust.city},{cust.state},
*             {cust.zip},{cust.area_code},{cust.phone})
* ::
*/
public int insertCustomerObject(Customer cust);

/**
 * <p>Demonstrates SQL DELETE statement using a multi-line @jc:sql tag.</p>
 *
 * @return The number of rows deleted.
 *
 * @param custid The key of the customer record to delete.
 *
 * @jc:sql statement::
 *     DELETE FROM customer WHERE custid={custid}
 * ::
 */
public int deleteCustomer(int custid);

/**
 * <p>Demonstrates a single-result SELECT query returning a <tt>Customer</tt> object.</p>
 *
 * @return A <tt>Customer</tt> object. Note that database column names map to the class's
 *         member names (case-insensitive).
 *
 * @param custid The key of the customer record.
 *
 * @jc:sql statement="SELECT custid, name, address, city, state, zip, area_code, phone FROM
 */
public Customer findCustomerByID(int key);

/**
 * <p>Demonstrates a single-row SELECT query returning a HashMap.</p>
 *
 * @return A HashMap containing the customer record. Note that the keys to the
 *         HashMap are the names of the database columns (upper-case names).
 *
 * @param custid The key of the customer record.
 *
 * @jc:sql statement="SELECT * FROM customer WHERE custid = {key}"
 */
public java.util.HashMap findCustomerHashByID(int key);

/**
 * <p>Demonstrates a single-row SELECT query returning a single column value.</p>
 *
 * @return The 'CITY' column of the customer record.
 *
 * @param custid The key of the customer record.
 *
 * @jc:sql statement="SELECT city FROM customer WHERE custid = {key}"
 */
public String findCustomerCityByID(int key);

/**
 * <p>Demonstrates a multiple-row SELECT query returning an Iterator.</p>
 *
 * @return A java.util.Iterator containing a
```

navWebServices.html Sample

```
*          <tt>CustomerDBControl.Customer</tt> object for
*          each row in the result set.
*
* @jc:sql statement="SELECT * FROM customer"
*       iterator-element-type="database.customer_db.Customer"
*/
public java.util.Iterator findAllCustomersIterator();

/**
 * <p>Demonstrates a multiple-result SELECT query returning an array.</p>
 *
 * @return An array containing a <tt>CustomerDBControl.Customer</tt>
 *         object for each row in the result set.
 *
 * @jc:sql statement="SELECT * FROM customer"
 *       array-max-length=100
 */
public Customer[] findAllCustomersArray();

/**
 * <p>Demonstrates SQL UPDATE statement.</p>
 *
 * <p>The return type for methods that use INSERT, UPDATE, or DELETE
 * must either be 'int' or 'void'.</p>
 *
 * @return The number of records updated.
 *
 * @param custid The key of the customer record.
 * @param address The new street address for the customer record.
 *
 * @jc:sql statement="UPDATE customer SET address = {address} WHERE custid={custid}"
 */
public int changeAddress(int custid, String address);

/**
 * @jc:sql statement="SELECT name FROM customer"
 *       array-max-length=100
 */
public String [] getAllCustomerNames();

/**
 * <p>Demonstrates a multiple-row SELECT query returning a ResultSet.</p>
 *
 * @return A java.sql.ResultSet containing all customer records.
 *
 * @jc:sql statement="SELECT * FROM customer"
 */
public java.sql.ResultSet findAllCustomersResultSet();

/**
 * <p>Demonstrates how to invoke an internal database function.
 * The function called is: in(row, rowIDs).
 *
 * @jc:sql statement="SELECT * FROM customer WHERE {sql:fn in(custid,{customerIDs})}"
 */
Customer[] callInternalFunction(Integer[] customerIDs);
}
```


CustomerDBClient.jws Sample

This topic includes the source code for the CustomerDBClient.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/database/customer_db/

Sample Source Code

```
package database.customer_db;

import java.sql.SQLException;
import java.sql.ResultSet;

/**
 * <p>A sample web service that exercises the CustomerDB.jcx Database control
 * extension. This sample uses the CUSTOMER database table that is created
 * and populated when WebLogic Workshop is installed.</p>
 *
 * @common:xmlns namespace="http://openuri.org/bea/samples/workshop/database" prefix="ns0"
 * @common:target-namespace namespace="http://workshop.bea.com/CustomerDBClient"
 */
public class CustomerDBClient implements com.bea.jws.WebService
{
    /**
     * <p>This is a Database control. It is defined in
     * CustomerDB.jcx. The @common:control tag
     * causes display as a control in the IDE's
     * Design View.</p>
     *
     * @common:control
     */
    private CustomerDB customerDB;

    /**
     * @common:operation
     */
    public String badSQL()
    {
        try
        {
            customerDB.badSQL();
        }
        catch (java.sql.SQLException e)
        {
            return "Failed: " + e.getMessage();
        }
        return "Created";
    }

    /**
     * <p>Use a database control method which inserts a row from individual arguments.</p>
     *
     */
}
```

navWebServices.html Sample

```
* @return The number of rows inserted.
*
* @param custid The key of the customer record.
* @param name The new name for the customer record.
* @param address The new street address for the customer record.
* @param city The new city for the customer record.
* @param state The new 2 letter state code for the customer record.
* @param zip The new 9 digit zip code for the customer record.
* @param area The new 3 digit area code for the customer record.
* @param phone The new 7 digit phone number for the customer record.
*
* @common:operation
*/
public int insertCustomer(int custid, String name, String address, String city, String state,
{
    int numInserted = customerDB.insertCustomer(custid, name, address, city, state, zip, area, phone);
    return numInserted;
}

/**
 * @common:operation
 */
public String [] getAllCustomerNamesArrayFromResultSet() throws Exception
{
    java.sql.ResultSet resultSet;
    String [] custNames;
    int numCustomers = customerDB.getAllCustomerNames().length;
    int i = 0;

    if( numCustomers > 0 )
    {
        custNames = new String [numCustomers];
        resultSet = customerDB.findAllCustomersResultSet();
        while (resultSet.next())
        {
            custNames[i++] = new String(resultSet.getString("name"));
        }
    }
    else
    {
        custNames = new String [1];
        custNames[0] = "no records found";
    }
    return custNames;
}

/**
 * @common:operation
 */
public String[] getAllCustomerNamesArray()
{
    return customerDB.getAllCustomerNames();
}

/**
 * <p>Use a database control method which returns all records,
 * using an Iterator internally.</p>
 *
 * @return Multiple customer records delimited by ". ".

```

navWebServices.html Sample

```
*
* @common:operation
*/
public String getAllCustomerNamesString()
{
    java.util.Iterator iter = null;
    StringBuffer sb = new StringBuffer();

    iter = customerDB.findAllCustomersIterator();
    if( iter.hasNext() )
    {
        while (iter.hasNext())
        {
            Customer c = (Customer)iter.next();
            sb.append( c.getCustid() ).append(':').append( c.getName() ).append(". ");
        }
    }
    else
    {
        sb.append("No records in table");
    }
    return sb.toString();
}

/**
 * <p>Use a database control method which returns all records,
 * using an array internally. Return that array directly
 * to the caller using the default XML mapping.</p>
 *
 * @common:operation
 */
public Customer [] getAllCustomerRecordsArray()
{
    Customer [] custArray;
    custArray = customerDB.findAllCustomersArray();
    return custArray;
}

/**
 * <p>Use a database control method to retrieve a customer record.</p>
 *
 * @return The customer's name. Return a message saying the customer
 *         is not found, if the customer is not in the table.
 *
 * @param custid The key of the customer record.
 *
 * @common:operation
 */
public String getCustomerName(int custid)
{
    Customer cust = null;
    cust = customerDB.findCustomerByID(custid);
    if( cust != null )
    {
        return cust.getName();
    }
    else
    {
        return("Customer not found");
    }
}
```

navWebServices.html Sample

```
}

/**
 * <p>Use a database control method to get the customer's street address.</p>
 *
 * <p>In this example, the customer record is returned in a HashMap, and then the
 * address is found by searching the HashMap with an UPPERCASE column name.
 * Database column names are always accessed in UPPERCASE in a HashMap.</p>
 *
 * @return The customer's street address. Return a message saying the customer
 *         is not found, if the customer is not in the table.
 *
 * @param custid The key of the customer record.
 *
 * @common:operation
 */
public String getCustomerStreetAddress(int custid)
{
    java.util.HashMap custHash = null;
    custHash = customerDB.findCustomerHashByID(custid);
    if (custHash != null)
    {
        return (String)custHash.get("ADDRESS");
    }
    else
    {
        return ("Customer not found");
    }
}

/**
 *
 * <p>Use a database control method to get the customer's full address.</p>
 *
 * <p>In this example, the customer record is returned in a Customer
 * object. The members of interest are then returned to the caller
 * in a custom XML map.</p>
 *
 * @return The city portion of the customer's record.
 *
 * @param custid The key of the customer record.
 *
 * @common:operation
 * @jws:return-xml schema-element="ns0:full-address" xquery::
 * declare namespace ns0="http://workshop.bea.com/CustomerDBClient"
 * declare namespace ns1="http://openuri.org/bea/samples/workshop/database"
 *
 * <ns1:full-address>
 *   <ns1:address>{data($input//ns0:address)}</ns1:address>
 *   <ns1:city>{data($input//ns0:city)}</ns1:city>
 *   <ns1:state>{data($input//ns0:state)}</ns1:state>
 *   <ns1:zip>{data($input//ns0:zip)}</ns1:zip>
 * </ns1:full-address>
 * ::
 */
public Customer getCustomerFullAddress(int custid)
{
    return customerDB.findCustomerByID(custid);
}
```

navWebServices.html Sample

```
/**
 * <p>Use a database control method which inserts a row from an object.</p>
 *
 * @return The number of rows inserted.
 *
 * @param custid The key of the customer record.
 * @param name The new name for the customer record.
 * @param address The new street address for the customer record.
 * @param city The new city for the customer record.
 * @param state The new 2 letter state code for the customer record.
 * @param zip The new 9 digit zip code for the customer record.
 * @param area The new 3 digit area code for the customer record.
 * @param phone The new 7 digit phone number for the customer record.
 *
 * @common:operation
 */
public int insertCustomerObj(int custid, String name, String address, String city, String s
{
    Customer custObj = new Customer( custid, name, address, city, state, zip,
                                     area, phone);
    int numInserted = customerDB.insertCustomerObject(custObj);
    return numInserted;
}

/**
 * <p>Use a database control method which updates a row.</p>
 *
 * @param custid The key of the customer record.
 * @param address The new address to be stored in the customer record.
 *
 * @common:operation
 */
public void updateAddress(int custid, String address)
    throws Exception
{
    int numUpdated = customerDB.changeAddress(custid, address);
    if (numUpdated != 1)
    {
        throw new Exception("Error updating address for customer " + custid + ".");
    }
}

/**
 * <p>Use a database control method that deletes a row.</p>
 *
 * @return The number of rows deleted.
 *
 * @param custid The key of the customer record to delete.
 *
 * @common:operation
 */
public int deleteCustomer(int custid)
{
    int numDeleted = customerDB.deleteCustomer(custid);
    return numDeleted;
}
```

navWebServices.html Sample

```
/**
 * <p>Demonstrates how to call an internal database function.</p>
 *
 * <p>Pass an array of integers when invoking the method. The
 * database will return an records with custid's within the
 * array of integers. For example, pass the array [1,2] or [2,3].</p>
 *
 * @return Customer[] object
 *
 * @param an array of integers, corresponding to custid's
 *
 * @common:operation
 */
public database.customer_db.Customer[] callInternalFunction(java.lang.Integer[] customerIDs)
{
    return customerDB.callInternalFunction(customerIDs);
}
}
```

dynamicSQL Samples

This section contains source code for the following samples.

Samples Included in This Section

DynamicSQL.jws Sample

ItemsDBControl.jcx Sample

DynamicSQL.jws Sample

This topic includes the source code for the DynamicSQL.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/database/dynamicSQL/

Sample Source Code

```
package database.dynamicSQL;

import database.dynamicSQL.ItemsDBControl.ItemRecord;
/**
 * This web service demonstrates the different sorts of dynamically generated SQL statements
 * and phrases that can be passed to a database control.
 */
public class DynamicSQL implements com.bea.jws.WebService
{
    /**
     * @common:control
     */
    private database.dynamicSQL.ItemsDBControl itemsDBControl;

    static final long serialVersionUID = 1L;

    /**
     * This method passes an entire SQL statement to the database control.
     *
     * The other methods of this web service pass SQL phrases to the database control.
     *
     * @common:operation
     */
    public ItemRecord[] dynamicStatement()
    {
        String queryParam = "SELECT * FROM WEBLOGIC.ITEMS WHERE ITEMNAME LIKE '%" + "cycle" + "
        return itemsDBControl.dynamicStatement(queryParam);
    }

    /**
     * This method passes a SQL WHERE phrase to the database control.
     *
     * @common:operation
     */
    public ItemRecord dynamicWhereClause()
    {
        return itemsDBControl.dynamicWhereClause("WHERE ITEMNUMBER = 624");
    }

    /**
     * This method passes a single value to the database control
     *
     * @common:operation
     */
    public ItemRecord simpleSubstitutionWhere()
```


navWebServices.html Sample

```
{
    return itemsDBControl.simpleSubstitutionWhere(Integer.parseInt("624"));
}

/**
 * This method passes a SQL LIKE phrase to the database control.
 *
 * @common:operation
 */
public ItemRecord[] dynamicLikeClause()
{
    return itemsDBControl.dynamicLikeClause("LIKE '%cycle%'");
}

/**
 * This method passes a pre-formatted string into a LIKE clause.
 * Note that the pre-formatted string does not include single quotes, because these are
 * implicitly added when the value is substituted into the LIKE clause.
 *
 * @common:operation
 */
public ItemRecord[] simpleSubstitutionLike()
{
    String paraMatchString = "%" + "cycle" + "%";
    return itemsDBControl.simpleSubstitutionLike(paraMatchString);
}
}
```

ItemsDBControl.jcx Sample

This topic includes the source code for the ItemsDBControl.jcx Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/database/dynamicSQL/

Sample Source Code

```
package database.dynamicSQL;

import com.bea.control.*;
import java.sql.SQLException;

/**
 * Defines a new database control.
 *
 * The @jc:connection tag indicates which WebLogic data source will be used by
 * this database control. Please change this to suit your needs. You can see a
 * list of available data sources by going to the WebLogic console in a browser
 * (typically http://localhost:7001/console) and clicking Services, JDBC,
 * Data Sources.
 *
 * @jc:connection data-source-jndi-name="cgSampleDataSource"
 */
public interface ItemsDBControl extends DatabaseControl, com.bea.control.ControlExtension
{
    // Add "throws SQLException" to request that SQLExeptions be thrown on errors.

    static public class ItemRecord
    {
        public int ItemNumber;
        public String ItemName;
        public int QuantityAvailable;
        public double Price;
    }

    static final long serialVersionUID = 1L;

    /**
     * This method accepts a whole SQL query as a substitution parameter.
     *
     * @jc:sql statement="{sql: query}"
     */
    ItemRecord[] dynamicStatement(String query);

    /**
     * This method accepts a SQL WHERE phrase as a substitution parameter.
     *
     * @jc:sql statement="SELECT * FROM WEBLOGIC.ITEMS {sql: whereClause}"
     */
    ItemRecord dynamicWhereClause(String whereClause);
}
```

navWebServices.html Sample

```
* In this method, the {sql: } substitution syntax is not necessary, since an individual
* value is being substituted into a WHERE clause.
*
* @jc:sql statement="SELECT * FROM WEBLOGIC.ITEMS WHERE ITEMNUMBER = {whereValue}"
*/
ItemRecord simpleSubstitutionWhere(int whereValue);

/**
 * This method accepts a SQL LIKE clause as a substitution parameter.
 *
 * @jc:sql statement="SELECT * FROM WEBLOGIC.ITEMS WHERE ITEMNAME {sql: likeClause}"
 */
ItemRecord[] dynamicLikeClause(String likeClause);

/**
 * In this method, the {sql: } substitution syntax is not necessary, since an individual
 * value is being substituted into a LIKE clause.
 *
 * Note that single quotes are added to the substituted value.
 *
 * The final SQL statement passed to the database is:
 * SELECT * FROM WEBLOGIC.ITEMS WHERE ITEMNAME LIKE '%CYCLE%'
 *
 * @jc:sql statement="SELECT * FROM WEBLOGIC.ITEMS WHERE ITEMNAME LIKE {likeValue}"
 */
ItemRecord[] simpleSubstitutionLike(String likeValue);

}
```

lucky_number_db Samples

This section contains source code for the following samples.

Samples Included in This Section

LuckyNumber.jws Sample

LuckyNumberControl.jcx Sample

LuckyNumberDB.jcx Sample

LuckyNumberDBClient.jws Sample

PlayerRecord.java Sample

LuckyNumber.jws Sample

This topic includes the source code for the LuckyNumber.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/database/lucky_number_db/

Sample Source Code

```
package database.lucky_number_db;

/**
 * <p>A simple web service that shows how to create an exposed method that
 * returns a value.</p>
 * @common:target-namespace namespace="http://workshop.bea.com/LuckyNumber"
 */
public class LuckyNumber implements com.bea.jws.WebService
{
    /**
     * <p>Returns a lucky number to the client.</p>
     *
     * @return A random number between 1 and 20.
     *
     * @common:operation
     */
    public int getLuckyNumber()
    {
        return 1 + (int)(20.0 * Math.random());
    }
}
```

LuckyNumberControl.jcx Sample

This topic includes the source code for the LuckyNumberControl.jcx Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/database/lucky_number_db/

Sample Source Code

```
package database.lucky_number_db;

/**
 * <p>A simple web service that shows how to create an exposed method that returns a value.</p>
 * @jc:location http-url="LuckyNumber.jws" jms-url="LuckyNumber.jws"
 * @jc:wSDL file="#LuckyNumberWSDL"
 * @editor-info:link autogen-style="java" source="LuckyNumber.jws" autogen="true"
 */
public interface LuckyNumberControl extends com.bea.control.ControlExtension, com.bea.control.S
{

    /**
     * <p>Returns a lucky number to the client.</p>
     */
    public int getLuckyNumber ();

    static final long serialVersionUID = 1L;
}

/** @common:define name="LuckyNumberWSDL" value::
<?xml version="1.0" encoding="utf-8"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:conv="http://www.openuri.org/20
<types>
    <s:schema elementFormDefault="qualified" targetNamespace="http://workshop.bea.com/Lucky
        <s:element name="getLuckyNumber">
            <s:complexType>
                <s:sequence/>
            </s:complexType>
        </s:element>
        <s:element name="getLuckyNumberResponse">
            <s:complexType>
                <s:sequence>
                    <s:element name="getLuckyNumberResult" type="s:int"/>
                </s:sequence>
            </s:complexType>
        </s:element>
        <s:element name="int" type="s:int"/>
    </s:schema>

</types>
<message name="getLuckyNumberSoapIn">
    <part name="parameters" element="s0:getLuckyNumber"/>
</message>
<message name="getLuckyNumberSoapOut">
    <part name="parameters" element="s0:getLuckyNumberResponse"/>
</message>
```

navWebServices.html Sample

```
</message>
<message name="getLuckyNumberHttpGetIn"/>
<message name="getLuckyNumberHttpGetOut">
  <part name="Body" element="s0:int"/>
</message>
<message name="getLuckyNumberHttpPostIn"/>
<message name="getLuckyNumberHttpPostOut">
  <part name="Body" element="s0:int"/>
</message>
<portType name="LuckyNumberSoap">
  <operation name="getLuckyNumber">
    <documentation><p>Returns a lucky number to the client.</p></documentation>
    <input message="s0:getLuckyNumberSoapIn"/>
    <output message="s0:getLuckyNumberSoapOut"/>
  </operation>
</portType>
<portType name="LuckyNumberHttpGet">
  <operation name="getLuckyNumber">
    <documentation><p>Returns a lucky number to the client.</p></documentation>
    <input message="s0:getLuckyNumberHttpGetIn"/>
    <output message="s0:getLuckyNumberHttpGetOut"/>
  </operation>
</portType>
<portType name="LuckyNumberHttpPost">
  <operation name="getLuckyNumber">
    <documentation><p>Returns a lucky number to the client.</p></documentation>
    <input message="s0:getLuckyNumberHttpPostIn"/>
    <output message="s0:getLuckyNumberHttpPostOut"/>
  </operation>
</portType>
<binding name="LuckyNumberSoap" type="s0:LuckyNumberSoap">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
  <operation name="getLuckyNumber">
    <soap:operation soapAction="http://workshop.bea.com/LuckyNumber/getLuckyNumber" style="rpc"/>
    <input>
      <soap:body use="literal"/>
    </input>
    <output>
      <soap:body use="literal"/>
    </output>
  </operation>
</binding>
<binding name="LuckyNumberHttpGet" type="s0:LuckyNumberHttpGet">
  <http:binding verb="GET"/>
  <operation name="getLuckyNumber">
    <http:operation location="/getLuckyNumber"/>
    <input>
      <http:urlEncoded/>
    </input>
    <output>
      <mime:mimeType part="Body"/>
    </output>
  </operation>
</binding>
<binding name="LuckyNumberHttpPost" type="s0:LuckyNumberHttpPost">
  <http:binding verb="POST"/>
  <operation name="getLuckyNumber">
    <http:operation location="/getLuckyNumber"/>
    <input>
      <mime:contentType="application/x-www-form-urlencoded"/>
    </input>
```

navWebServices.html Sample

```
<output>
  <mime:mimeXml part="Body"/>
</output>
</operation>
</binding>
<service name="LuckyNumber">
  <documentation>&lt;p>A simple web service that shows how to create an exposed method</p>
  <port name="LuckyNumberSoap" binding="s0:LuckyNumberSoap">
    <soap:address location="http://localhost:7001/database/LuckyNumber.jws"/>
  </port>
  <port name="LuckyNumberHttpGet" binding="s0:LuckyNumberHttpGet">
    <http:address location="http://localhost:7001/database/LuckyNumber.jws"/>
  </port>
  <port name="LuckyNumberHttpPost" binding="s0:LuckyNumberHttpPost">
    <http:address location="http://localhost:7001/database/LuckyNumber.jws"/>
  </port>
</service>
</definitions>
* ::
*/
```


LuckyNumberDB.jcx Sample

This topic includes the source code for the LuckyNumberDB.jcx Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/database/lucky_number_db/

Sample Source Code

```
package database.lucky_number_db;

import com.bea.control.DatabaseControl;
import com.bea.control.ControlException;
import java.sql.SQLException;
import java.util.Iterator;

/**
 * Defines a new database control.
 *
 * The @jc:connection tag indicates which WebLogic data source will be used by
 * this database control. Please change this to suit your needs. You can see a
 * list of available data sources by going to the WebLogic console in a browser
 * (typically http://localhost:7001/console) and clicking Services, JDBC,
 * Data Sources.
 *
 * @jc:connection data-source-jndi-name="cgSampleDataSource"
 */
public interface LuckyNumberDB extends com.bea.control.ControlExtension, DatabaseControl
{

    /**
     * <p>Creates the PLAYERS table with columns NAME and NUMBER.</p>
     *
     * <p>The PLAYERS table is created and populated when WebLogic Workshop
     * is installed. This method is here to illustrate that CREATE TABLE, like
     * all SQL commands, can be issued from a Database control method.</p>
     *
     * @jc:sql statement="CREATE TABLE PLAYERS ( NAME VARCHAR(30), NUMBER INT )"
     *
     * @throws java.sql.SQLException
     */
    void createTable() throws SQLException;

    /**
     * <p>Adds a player (NAME, NUMBER) to the PLAYERS table.</p>
     *
     * @jc:sql statement="INSERT INTO PLAYERS (NAME, NUMBER) VALUES ({playerName},{number})"
     *
     * @throws java.sql.SQLException
     */
    void insertPlayer(String playerName, int number) throws SQLException;

}
```

navWebServices.html Sample

```
* <p>Returns an Iterator containing a PlayerRecord for each player in the PLAYERS table.</p>
*
* <p>Note that in the Alpha release, you must not allow execution of additional
* queries until you are finished using the Iterator used by this method.</p>
*
* @jc:sql statement="SELECT NAME, NUMBER FROM PLAYERS ORDER BY NUMBER"
*       iterator-element-type="database.lucky_number_db.PlayerRecord"
*
* @throws java.sql.SQLException
* @throws com.bea.control.ControlException
*/
Iterator getAllPlayers() throws SQLException, ControlException;

/**
* <p>Returns an Iterator containing a PlayerRecord for each player whose NUMBER=winningNum
*
* <p>Note that in the Alpha release, you must not allow execution of additional
* queries until you are finished using the Iterator used by this method.</p>
*
* @jc:sql statement="SELECT NAME, NUMBER FROM PLAYERS WHERE NUMBER = {winningNumber} ORDER
*       iterator-element-type="database.lucky_number_db.PlayerRecord"
*
* @throws java.sql.SQLException
* @throws com.bea.control.ControlException
*/
Iterator getAllWinners(int winningNumber) throws SQLException, ControlException;

/**
* <p>Deletes the PLAYERS table from the database.</p>
*
* <p>The PLAYERS table is created and populated when WebLogic Workshop
* is installed. This method is here to illustrate that DROP TABLE, like
* all SQL commands, can be issued from a Database control method.</p>
*
* @jc:sql statement="DROP TABLE PLAYERS"
*
* @throws java.sql.SQLException
*/
void destroyTable() throws SQLException;
}
```

LuckyNumberDBClient.jws Sample

This topic includes the source code for the LuckyNumberDBClient.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/database/lucky_number_db/

Sample Source Code

```
package database.lucky_number_db;

import java.sql.SQLException;
import java.util.Iterator;
import database.lucky_number_db.LuckyNumberControl;
import database.lucky_number_db.LuckyNumberDB;
import com.bea.control.JwsContext;

/**
 * <p>A sample web service that exercises the CustomerDB.jcx Database control
 * extension. This sample uses the PLAYERS database table that is created
 * and populated when WebLogic Workshop is installed.</p>
 * @common:target-namespace namespace="http://workshop.bea.com/LuckyNumberDBClient"
 */
public class LuckyNumberDBClient implements com.bea.jws.WebService
{
    /**
     * @common:context
     */
    JwsContext context;

    /**
     * @common:control
     */
    private database.lucky_number_db.LuckyNumberControl luckyNumber;

    /**
     * @common:control
     */
    private LuckyNumberDB luckyNumberDB;

    /**
     * <p>Starts a conversation.</p>
     *
     * <p>The default players have lucky numbers in the range [1,20] inclusive.</p>
     *
     * @common:operation
     * @jws:conversation phase="start"
     */
    public void start() throws SQLException
    {
        return;
    }
}
```

navWebServices.html Sample

```
* <p>Gets a lucky number from LuckyNumber.jws, then returns a string containing
* the number and the list of winners, if any.</p>
*
* @common:operation
* @jws:conversation phase="continue"
*/
public String drawNumberAndGetWinner() throws Exception
{
    return listWinnersForNumber(luckyNumber.getLuckyNumber());
}

/**
* <p>Returns a list of all players in the database.</p>
*
* <p>Players's names and lucky numbers are returned in a string, one player per line.</p>
*
* @common:operation
* @jws:conversation phase="continue"
*/
public String listPlayers() throws Exception
{
    StringBuffer sb = new StringBuffer("\nPlayers:\n");
    Iterator iter = luckyNumberDB.getAllPlayers();

    if (!iter.hasNext())
    {
        sb.append("None");
    }
    while (iter.hasNext())
    {
        PlayerRecord rec =
            (PlayerRecord)iter.next();
        sb.append("Number: ");
        sb.append(rec.number);
        sb.append(", Name: ");
        sb.append(rec.name);
        sb.append("\n");
    }
    return sb.toString();
}

/**
* <p>Returns a string containing the number and the list of winners, if any.</p>
*
* @param luckyNumber Winning number for which winners will be returned.
* @return A string containing the number and the list of winners, if any.
*
* @common:operation
* @jws:conversation phase="continue"
*/
public String listWinnersForNumber(int luckyNumber) throws Exception
{
    StringBuffer sb = new StringBuffer("Number: " + luckyNumber + "; Winners: ");
    Iterator iter = luckyNumberDB.getAllWinners(luckyNumber);
    if (!iter.hasNext())
    {
        sb.append("None");
    }
    while (iter.hasNext())
    {
        PlayerRecord rec =
```

navWebServices.html Sample

```
        (PlayerRecord)iter.next();
        sb.append(rec.name);
        if (iter.hasNext())
        {
            sb.append(", ");
        }
    }
    return sb.toString();
}

/**
 * <p>Terminates the client conversation.</p>
 *
 * <p>This method should be called when the client is finished using the web service.</p>
 *
 * @common:operation
 * @jws:conversation phase="finish"
 */
public void end()
{
    return;
}

}
```

PlayerRecord.java Sample

This topic includes the source code for the PlayerRecord.java Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/database/lucky_number_db/

Sample Source Code

```
package database.lucky_number_db;

/**
 * <p>Utility class used in the LuckyNumberDB.jcx and LuckyNumber.jws samples.
 * The names of the members of this class must match the column names and types
 * in the database table, with the exception of case.</p>
 */
public class PlayerRecord
{
    public String name;
    public int number;
}
```

xmlBean Samples

This section contains source code for the following samples.

Samples Included in This Section

CustomerDB_XMLBean.jcx Sample

CustomerDB_XMLBeanClient.jws Sample

ItemsDB_XMLBean.jcx Sample

ItemsDB_XMLBean_Client.jws Sample

CustomerDB_XMLBean.jcx Sample

This topic includes the source code for the CustomerDB_XMLBean.jcx Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/database/xmlBean/

Sample Source Code

```
package database.xmlBean;

import databaseCustomerDb.XCustomerDocument;
import databaseCustomerDb.XCustomerDocument.XCustomer;
import databaseCustomerDb.XCustomerDocument.Factory;
import com.bea.control.DatabaseControl;
import databaseCustomerDb.XCustomerDocument.XCustomer.XCustomerRow;
import java.sql.SQLException;

/**
 * <p> A sample Database Control that demonstrates how to
 * load database data into XMLBean types.</p>
 *
 * <p>This Database Control uses the JDBC Data Source
 * "cgSampleSataSource" which is configured automatically
 * when WebLogic Workshop is installed.</p>
 *
 * <p>The data source is configured with the JNDI
 * name "cgSampleDataSource".</p>
 *
 * <p>The SQL queries refer to the CUSTOMER table,
 * which is pre-loaded in the Pointbase database.</p>
 *
 * <p>The CUSTOMER table contains the following fields:
 *
 *   <ul>
 *     <li>CUSTID (INT, Primary Key)</li>
 *     <li>NAME (VARCHAR)</li>
 *     <li>ADDRESS (VARCHAR)</li>
 *     <li>CITY (VARCHAR)</li>
 *     <li>STATE (CHAR(2))</li>
 *     <li>ZIP (VARCHAR)</li>
 *     <li>AREA_CODE (CHAR(3))</li>
 *     <li>PHONE CHAR(9)</li>
 *   </ul>
 *
 * </p>
 *
 * @jc:connection data-source-jndi-name="cgSampleDataSource"
 */

public interface CustomerDB_XMLBean extends com.bea.control.ControlExtension, DatabaseControl
{
    /**
     * @jc:sql statement="SELECT * FROM customer"
     */
}
```


navWebServices.html Sample

```
public XCustomerDocument findAllCustomersDoc();

/**
 * @jc:sql statement="SELECT name FROM customer"
 *       array-max-length=100
 */
public XCustomerDocument getAllCustomerNames();

/**
 * @jc:sql statement="SELECT custid, name, address, city, state, zip, area_code, phone FROM
 */
public XCustomerDocument getCustomerByID(int key);

/**
 * @jc:sql statement="SELECT city FROM customer WHERE custid = {key}"
 */
public XCustomerDocument getCustomerCityByID(int key);

/**
 * This method takes an XCustomerRow object
 *
 *      <XCustomerRow>
 *          <CUSTID>1111</CUSTID>
 *          <NAME>Some Name</NAME>
 *          <ADDRESS>Some Address</ADDRESS>
 *          <CITY>Some City</CITY>
 *          <STATE>Some State</STATE>
 *          <ZIP>00000</ZIP>
 *          <AREA_CODE>111</AREA_CODE>
 *          <PHONE>123-4567</PHONE>
 *      </XCustomerRow>
 *
 * and parses it into a SQL INSERT statement before passing the statement onto the database
 * Note that the substitution syntax uses the XML document's elements names: {cust.CUSTID}
 *
 * @jc:sql statement::
 *      INSERT INTO customer (custid,name,address,city,state,zip,area_code,phone)
 *      VALUES ({cust.CUSTID},{cust.NAME},{cust.ADDRESS},{cust.CITY},{cust.STATE},
 *              {cust.ZIP},{cust.AREACODE},{cust.PHONE})
 *      ::
 */
public int insertCustomerBean(XCustomerRow cust);
}
```

CustomerDB_XMLBeanClient.jws Sample

This topic includes the source code for the CustomerDB_XMLBeanClient.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/database/xmlBean/

Sample Source Code

```
package database.xmlBean;

import databaseCustomerDb.XCustomerDocument;
import databaseCustomerDb.XCustomerDocument.XCustomer;
import databaseCustomerDb.XCustomerDocument.XCustomer.XCustomerRow;
import java.io.File;

/**
 * <p>This web service is a client to the database control CustomerDB_XMLBean.jcx.
 *
 * <p>The methods of this web service show how XMLBeans types can be passed to and received from
 */
public class CustomerDB_XMLBeanClient implements com.bea.jws.WebService
{
    static final long serialVersionUID = 1L;

    /**
     * @common:control
     */
    public database.xmlBean.CustomerDB_XMLBean customerDB_XMLBean;

    /**
     * Returns an XML document containing all of the data in the CUSTOMER table.
     *
     * @common:operation
     */
    public XCustomerDocument findAllCustomersDoc()
    {
        return customerDB_XMLBean.findAllCustomersDoc();
    }

    /**
     * Returns an XML document containing all of the names in the CUSTOMER table.
     *
     * @common:operation
     */
    public XCustomerDocument getAllCustomerNames()
    {
        return customerDB_XMLBean.getAllCustomerNames();
    }

    /**
     * Returns an XML document containing one customer.
     *
     * @common:operation
     */
}
```

navWebServices.html Sample

```

    */
public XCustomerDocument getCustomerByID(int key)
{
    return customerDB_XMLBean.getCustomerByID(key);
}

/**
 * Returns an XML document containing the requested customer's city.
 *
 * @common:operation
 */
public XCustomerDocument getCustomerCityByID(int key)
{
    return customerDB_XMLBean.getCustomerCityByID(key);
}

/**
 * Shows how to construct a XCustomerDocument instance and pass this to the database control
 * The control method parses the XCustomerDocument into a SQL INSERT statement and passes t
 * on to the database.
 *
 * @common:operation
 */
public int insertCustomerBean (int custid, java.lang.String name, java.lang.String address,
    java.lang.String city, java.lang.String state, java.lang.String zip,
    java.lang.String area_code, java.lang.String phone)
{
    XCustomerDocument custDoc = null;
    XCustomer cust = null;
    XCustomerRow custRow = null;

    /*
     * Create an instance XML document.  XMLBean "Document" types represent an entire XML d
     */
    custDoc = XCustomerDocument.Factory.newInstance();
    /*
     * Add a <XCustomer> element to the document.
     */
    cust = custDoc.addNewXCustomer();
    /*
     * Add a <XCustomerRow> element to the document
     */
    custRow = cust.addNewXCustomerRow();
    /*
     * Add <CUSTID>, <NAME>, etc. elements to the document and set their values.
     */
    custRow.setCUSTID(custid);
    custRow.setName(name);
    custRow.setADDRESS(address);
    custRow.setCITY(city);
    custRow.setSTATE(state);
    custRow.setZIP(zip);
    custRow.setAREACODE(area_code);
    custRow.setPHONE(phone);

    /*
     * The entire document now looks like this:
     *
     * <!DOCTYPE XCustomer SYSTEM "">
     * <XCustomer xmlns="java:///database/customer_db" xmlns:xsi="http://www.w3.org/2001/X
     *     <XCustomerRow>

```

navWebServices.html Sample

```
*      <CUSTID>1111</CUSTID>
*      <NAME>Some Name</NAME>
*      <ADDRESS>Some Address</ADDRESS>
*      <CITY>Some City</CITY>
*      <STATE>Some State</STATE>
*      <ZIP>00000</ZIP>
*      <AREA_CODE>111</AREA_CODE>
*      <PHONE>123-4567</PHONE>
*      </XCustomerRow>
*    </XCustomer>
*
*    But only the <XCustomerRow> element is sent to the database control method.
*
*/
return customerDB_XMLBean.insertCustomerBean(custDoc.getXCustomer().getXCustomerRowArra
}
}
```

ItemsDB_XMLBean.jcx Sample

This topic includes the source code for the ItemsDB_XMLBean.jcx Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/database/xmlBean/

Sample Source Code

```
package database.xmlBean;

import com.bea.control.*;
import java.sql.SQLException;
import org.openuri.dbcontrol.xmlbean.ItemType;
import org.openuri.dbcontrol.xmlbean.ItemsArrayDocument;

/**
 * This database control returns XMLBean types based on the schema ItemsArray.xsd.
 *
 * See the the Schemas folder for the schema file ItemsArray.xsd and
 * its associated XMLBean types.
 *
 * @jc:connection data-source-jndi-name="cgSampleDataSource"
 */
public interface ItemsDB_XMLBean extends DatabaseControl, com.bea.control.ControlExtension
{

    static final long serialVersionUID = 1L;

    /**
     * @jc:sql statement="SELECT * FROM ITEMS"
     */
    ItemsArrayDocument getAllItems() throws SQLException;

    /**
     * @jc:sql statement="SELECT * FROM ITEMS WHERE ITEMNUMBER = {itemNumber}"
     */
    ItemType getIndividualItem(int itemNumber) throws SQLException;

}
```

ItemsDB_XMLBean_Client.jws Sample

This topic includes the source code for the ItemsDB_XMLBean_Client.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/database/xmlBean/

Sample Source Code

```
package database.xmlBean;

import java.sql.SQLException;
import org.openuri.dbcontrol.xmlbean.ItemsArrayDocument;

/**
 *
 * <p>The methods of the this web service return XMLBean datatypes.
 *
 * <p>The XMLBean datatypes ItemsArrayDocument and ItemType were generated from the schema file
 * ItemsArray.xsd (in the Schemas project).
 *
 * @common:target-namespace namespace="http://workshop.bea.com/ItemsWebService"
 */
public class ItemsDB_XMLBean_Client implements com.bea.jws.WebService
{
    /**
     * @common:control
     */
    private database.xmlBean.ItemsDB_XMLBean items;

    static final long serialVersionUID = 1L;

    /**
     * <p>The following method returns an ItemsArrayDocument type,
     * an XMLBean type based on the schema ItemsArray.xsd.
     *
     * <p>See the Schemas folder for the schema ItemsArray.xsd, and its corresponding XMLBean c
     *
     * @common:operation
     */
    public ItemsArrayDocument getAllItems()
        throws SQLException
    {
        return items.getAllItems();
    }

    /**
     * <p>The following method returns an ItemType type,
     * an XMLBean type based on the schema ItemsArray.xsd.
     *
     * <p>See the Schemas folder for the schema ItemsArray.xsd, and its corresponding XMLBean c
     *
     * <p>Enter an integer between 624 and 664.

```

navWebServices.html Sample

```
*
* @common:operation
*/
public org.openuri.dbcontrol.xmlbean.ItemType getIndividualItem(int itemNumber) throws java
{
    return items.getIndividualItem(itemNumber);
}
}
```

ejbControl Samples

This section contains source code for the following samples.

Samples Included in This Section

AccountEJBClient.jws Sample

AccountEJBControl.jcx Sample

TraderEJBClient.jws Sample

TraderEJBControl.jcx Sample

AccountEJBClient.jws Sample

This topic includes the source code for the AccountEJBClient.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/ejbControl/

Sample Source Code

```
package ejbControl;

import examples.ejb20.basic.containerManaged.Account;
import examples.ejb20.basic.containerManaged.ProcessingErrorException;
import javax.ejb.FinderException;
import javax.ejb.CreateException;
import java.rmi.RemoteException;
import java.util.Iterator;

/**
 * <p>A web service that demonstrates use of an EJB control AccountEJBControl.jcx,
 * which represents the AccountEJB Entity Bean and exposes its business interface
 * to web services.</p>
 * @common:target-namespace namespace="http://workshop.bea.com/AccountEJBClient"
 */
public class AccountEJBClient implements com.bea.jws.WebService
{
    /**
     * @common:control
     */
    private ejbControl.AccountEJBControl account;

    /**
     * <p>Invokes the target EJB's <b>create</b> method and returns the result.</p>
     *
     * <ul>
     * <li><b>key</b> is the account identifier (must be unique for each account).</li>
     * <li><b>openingBalance</b> is the starting balance of the account.</li>
     * <li><b>type</b> is the account type, e.g. "checking" or "savings".</li>
     * </ul>
     *
     * @return A String describing the result of the account creation.
     *
     * @common:operation
     */
    public String createNewAccount(String key, double openingBalance, String type)
    {
        try
        {
            account.create(key, openingBalance, type);
        }
        catch (CreateException ce)
        {
            return "Error " + ce.getLocalizedMessage();
        }
    }
}
```

navWebServices.html Sample

```
        catch (RemoteException re)
        {
            return "Error " + re.getLocalizedMessage();
        }
        return "Successful";
    }

    /**
     * <p>Invokes the target EJB's <b>getAccounts</b> method and returns a list of all
     * accounts currently persisted by the target EJB.</p>
     *
     * @return A String listing all of the accounts.
     *
     * @common:operation
     */
    public String listAccounts()
    {
        return getAccounts(0.0d);
    }

    /**
     * <p>Invokes the target EJB's <b>getAccounts</b> method and returns a list of accounts
     * with balances greater than the specified <b>threshold</b>.</p>
     *
     * @return A String listing all of the accounts.
     *
     * @common:operation
     */
    public String listBigAccounts(double threshold)
    {
        return getAccounts(threshold);
    }

    /**
     * <p>Invokes the target EJB's <b>getAccount</b> method and returns the account type of
     * the account specified by <b>accountKey</b>.</p>
     *
     * @return A String account type.
     *
     * @common:operation
     */
    public String accountType(String accountKey)
    {
        try
        {
            return getAccount(accountKey).accountType();
        }
        catch (FinderException fe)
        {
            return "Error " + fe.getLocalizedMessage();
        }
        catch (RemoteException re)
        {
            return "Error " + re.getLocalizedMessage();
        }
    }

    /**
     * <p>Invokes the target EJB's <b>getAccount</b> on the account specified by
     * <b>accountKey</b>, then the returned Account's <b>balance</b>
     * method. Returns the account balance.</p>
     */
```

navWebServices.html Sample

```
*
* @return A double account balance, or -1 on a Finder Exception or -2 on Remote Exception.
*
* @common:operation
*/
public double balance(String accountKey)
{
    try
    {
        return getAccount(accountKey).balance();
    }
    catch (FinderException fe)
    {
        return -1d;
    }
    catch (RemoteException re)
    {
        return -2d;
    }
}

/**
 * <p>Invokes the target EJB's <b>getAccount</b> on the account specified by
 * <b>accountKey</b>, then the returned Account's <b>deposit</b>
 * method. Returns the resulting account balance.</p>
 *
 * @return A double account balance, or -1 on a Finder Exception or -2 on Remote Exception.
 *
 * @common:operation
 */
public double deposit(String accountKey, double depositAmount)
{
    try
    {
        return getAccount(accountKey).deposit(depositAmount);
    }
    catch (FinderException fe)
    {
        return -1d;
    }
    catch (RemoteException re)
    {
        return -2d;
    }
}

/**
 * <p>Invokes the target EJB's <b>getAccount</b> on the account specified by
 * <b>accountKey</b>, then the returned account's <b>withdraw</b>
 * method. Returns the resulting account balance.</p>
 *
 * @return A double account balance,
 * or -1 on a Finder Exception,
 * or -2 on Remote Exception,
 * or -3 on Processing Error Exception.
 *
 * @common:operation
 */
public double withdraw(String accountKey, double withdrawAmount)
{
    try
```

navWebServices.html Sample

```

    {
        return getAccount(accountKey).withdraw(withdrawAmount);
    }
    catch (FinderException fe)
    {
        return -1d;
    }
    catch (RemoteException re)
    {
        return -2d;
    }
    catch (ProcessingErrorException pe)
    {
        return -3d;
    }
}

/**
 * <p>Helper function -- returns an EJB account for a given key<p>
 *
 * @return An Account object
 */
private Account getAccount(String accountKey)
    throws FinderException, RemoteException
{
    return account.findByPrimaryKey(accountKey);
}

/**
 * <p>Helper function -- formats account list returns<p>
 *
 * @return new line delimited string of accounts
 */
private String getAccounts(double limit)
{
    Iterator itr = null;
    StringBuffer retval = new StringBuffer();
    try
    {
        itr = account.findBigAccounts(limit).iterator();
        while (itr.hasNext())
        {
            Account a = (Account)itr.next();
            retval.append(a.getPrimaryKey()).append(' ').append(a.accountType()).append(' ');
        }
    }
    catch (FinderException fe)
    {
        return "Error " + fe.getLocalizedMessage();
    }
    catch (RemoteException re)
    {
        return "Error " + re.getLocalizedMessage();
    }

    return retval.toString();
}
}

```

AccountEJBControl.jcx Sample

This topic includes the source code for the AccountEJBControl.jcx Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/ejbControl/

Sample Source Code

```
package ejbControl;

/**
 * @jc:ejb home-jndi-name="ejb20-containerManaged-AccountHome"
 * @editor-info:ejb home="AccountEJB.jar" bean="AccountEJB.jar"
 */
public interface AccountEJBControl
    extends examples.ejb20.basic.containerManaged.AccountHome,    // home interface
           examples.ejb20.basic.containerManaged.Account,        // bean interface
           com.bea.control.EntityEJBControl,                      // control interface
           com.bea.control.ControlExtension                       // control extension marker
{
}
```

TraderEJBClient.jws Sample

This topic includes the source code for the TraderEJBClient.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/ejbControl/

Sample Source Code

```
package ejbControl;

import java.rmi.RemoteException;
import examples.ejb20.basic.statelessSession.TradeResult;

/**
 * <p>A web service that demonstrates use of the EJB control TraderEJBControl.jcx,
 * which represents the TraderEJB Stateless Session Bean and exposes its business
 * interface to web services.</p>
 * @common:target-namespace namespace="http://workshop.bea.com/TraderEJBClient"
 */
public class TraderEJBClient implements com.bea.jws.WebService
{
    /**
     * @common:control
     */
    private ejbControl.TraderEJBControl trader;

    /**
     * <p>Invokes the target EJB's <b>buy</b> method and returns the result.
     * Trades are restricted to 500 shares.</p>
     *
     * @return A String describing the result of the transaction.
     *
     * @common:operation
     */
    public String buy(String tickerSymbol, int numberOfShares)
    {
        TradeResult tr;
        try
        {
            tr = trader.buy(tickerSymbol, numberOfShares);
        }
        catch (RemoteException re)
        {
            return "Error " + re.getLocalizedMessage();
        }
        return String.valueOf(tr.getNumberTraded()) + " shares of " + tr.getStockSymbol() + " b

    }

    /**
     * <p>Invokes the target EJB's <b>sell</b> method and returns the result.
     * Trades are restricted to 500 shares.</p>
     *
     * @return A String describing the result of the transaction.
     */
}
```

navWebServices.html Sample

```
*
* @common:operation
*/
public String sell(String tickerSymbol, int numberOfShares)
{
    TradeResult tr;
    try
    {
        tr = trader.sell(tickerSymbol, numberOfShares);
    }
    catch (RemoteException re)
    {
        return "Error " + re.getLocalizedMessage();
    }
    return String.valueOf(tr.getNumberTraded()) + " shares of " + tr.getStockSymbol() + " s
}
}
```

TraderEJBControl.jcx Sample

This topic includes the source code for the TraderEJBControl.jcx Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/ejbControl/

Sample Source Code

```
package ejbControl;

/**
 * @jc:ejb home-jndi-name="ejb20-statelessSession-TraderHome"
 * @editor-info:ejb home="TraderEJB.jar" bean="TraderEJB.jar"
 */
public interface TraderEJBControl
    extends examples.ejb20.basic.statelessSession.TraderHome,    // home interface
           examples.ejb20.basic.statelessSession.Trader,        // bean interface
           com.bea.control.SessionEJBControl,                   // control interface
           com.bea.control.ControlExtension                     // control extension marker
{
}
```


Error.jsp Sample

This topic includes the source code for the Error.jsp Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/

Sample Source Code

```
<!--Generated by WebLogic Workshop-->
<%@ page language="java" contentType="text/html; charset=UTF-8" isErrorPage="true"%>
<%@ taglib uri="netui-tags-databinding.tld" prefix="netui-data"%>
<%@ taglib uri="netui-tags-html.tld" prefix="netui"%>
<%@ taglib uri="netui-tags-template.tld" prefix="netui-template"%>
<netui:html>
  <head>
    <title>An Error Occurred</title>
    <link href="resources/css/style.css" type="text/css" rel="stylesheet"/>
    <netui:base/>
  </head>
  <body bgcolor=white style="margin:0">
    <jsp:include page="/resources/jsp/header.jsp"/>
    <br/>
    <p>
      An error has occurred:
    </p>
    <blockquote>
      <netui:label value="{request.errorMessage}" defaultValue="" />
      <br/>
      <netui:exceptions showMessage="true" />

      <p>&nbsp;</p>

      <p>Please return to the Web Services Samples Home page.</p>

      <p>
        <netui:anchor href="Controller.jpj">Home</netui:anchor>
      </p>

    </blockquote>
  </body>
</netui:html>

<!-- Some browsers will not display this page unless the response status code is 200. -->
<% response.setStatus(200); %>
```

HelloWorld.jws Sample

This topic includes the source code for the HelloWorld.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/

Sample Source Code

```
/**
 * A very simple web service.
 *
 * @common:target-namespace namespace="http://workshop.bea.com/HelloWorld"
 */
public class HelloWorld implements com.bea.jws.WebService
{
    /**
     * Demonstrates what a web service method looks
     * like in its simplest form.
     *
     * This is a synchronous method, meaning
     * that a client of this web service will block
     * until this method returns. In this case,
     * the method doesn't do any significant work
     * so that is OK.
     *
     * The @returns tag is an example of a standard
     * Javadoc tag. Other typical tags you might
     * add to a method are @param, @exception, etc.
     *
     * The @common:operation tag is a Workshop-specific
     * Javadoc tag that marks this method as an
     * exposed method of the web service.
     *
     * @return 'Hello, World!', of course.
     *
     * @common:operation
     */
    public String Hello()
    {
        return "Hello, World!";
    }
}
```

Index.jsp Sample

This topic includes the source code for the Index.jsp Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/

Sample Source Code

```
<%@ page language="java" contentType="text/html; charset=UTF-8"%>

<%@ taglib uri="netui-tags-html.tld" prefix="netui" %>

<html>
<head>
<title>Web Services Feature Samples</title>
<link href="resources/css/style.css" type="text/css" rel="stylesheet" />
<netui:base/>
</head>

<body style="margin:0">
<jsp:include page="/resources/jsp/header.jsp"/>
<br/>

<table width="700" align="left" border="0">
  <tr>
    <td colspan="4">
      <p>This page provides links to samples that demonstrate web services
        created in WebLogic Workshop. You can find the files that comprise these
        samples in &lt;WEBLOGIC_HOME&gt;/samples/workshop/SamplesApp/WebServices/...</p>
      <p>&nbsp;</p>
    </td>
  </tr>
  <tr valign="top">
    <td width="25">&nbsp;</td>
    <td width="120" valign="top">
      <netui:image align="right" src="resources/images/samp3.jpg"
        width="119" height="133" vspace="10" border="0"/>
    </td>
    <td width="5" valign="top"></td>
    <td valign="top">
      <hr size="1" align="left">
      <p><b>Simple web service sample</b></p>
      <p>
        <a href="http://localhost:7001/WebServices/HelloWorld.jws?.EXPLORE=.TEST">HelloWorld.jws</a>
      </p>
      <p>Sources: /WebServices</p>
    </td>
  </tr>
  <tr valign="top">
    <td width="25">&nbsp;</td>
    <td width="120" valign="top">&nbsp;</td>
    <td width="5">&nbsp;</td>
    <td valign="top">
```

Index.jsp Sample

106

navWebServices.html Sample

```

<td width="5">&nbsp;</td>
<td valign="top">
  <hr size="1" align="left">
  <p><b>EJB control samples</b></p>
  <p>
    <a href="http://localhost:7001/WebServices/ejbControl/AccountEJBClient.jws?.EXPLORE=.TEST">AccountEJBClient.jws</a>
    <a href="http://localhost:7001/WebServices/ejbControl/TraderEJBClient.jws?.EXPLORE=.TEST">TraderEJBClient.jws</a>
  </p>
  <p>Sources: /WebServices/ejbControl</p>
</td>
</tr>

<tr valign="top">
  <td width="25">&nbsp;</td>
  <td width="120" valign="top">&nbsp;</td>
  <td width="5">&nbsp;</td>
  <td valign="top">
    <hr size="1" align="left">
    <p><b>.NET interoperability sample</b></p>
    <p><a href="http://localhost:7001/WebServices/interop/dotNET/readme.html?.EXPLORE=.TEST">dotNET/readme.html</a>
    </p>
    <p>Sources: /WebServices/interop/dotNET/ConversationClient.asmx.cs</p>
  </td>
</tr>

<tr>
  <td width="25">&nbsp;</td>
  <td width="120" valign="top">&nbsp;</td>
  <td width="5">&nbsp;</td>
  <td valign="top">
    <hr size="1" align="left">
    <p><b>JMS control samples</b></p>
    <p>
      <a href="http://localhost:7001/WebServices/jms/AccountPublish.jws?.EXPLORE=.TEST">AccountPublish.jws</a>
      <a href="http://localhost:7001/WebServices/jms/AccountSubscribe.jws?.EXPLORE=.TEST">AccountSubscribe.jws</a>
      <a href="http://localhost:7001/WebServices/jms/CustomJMSClient.jws?.EXPLORE=.TEST">CustomJMSClient.jws</a>
      <a href="http://localhost:7001/WebServices/jms/SimpleJMS.jws?.EXPLORE=.TEST">SimpleJMS.jws</a>
      <a href="http://localhost:7001/WebServices/jms_xmlProtocol/JMS_XMLProtocol.jws?.EXPLORE=.TEST">JMS_XMLProtocol.jws</a>
    </p>
    <p>Sources: /WebServices/jms</p>
  </td>
</tr>

<tr valign="top">
  <td width="25">&nbsp;</td>
  <td width="120" valign="top">&nbsp;</td>
  <td width="5">&nbsp;</td>
  <td valign="top">
    <hr size="1" align="left">
    <p><b>Inline custom control samples</b></p>
    <p>
      <a href="http://localhost:7001/WebServices/localControls/LocalControlTest.jws?.EXPLORE=.TEST">LocalControlTest.jws</a>
    </p>
    <p>Sources: /WebServices/localControls</p>
  </td>
</tr>

<tr>
  <td width="25">&nbsp;</td>
  <td width="120" valign="top">&nbsp;</td>
  <td width="5">&nbsp;</td>

```

navWebServices.html Sample

```

<td valign="top">
  <hr size="1" align="left">
  <p><b>Proxy web service samples</b></p>
  <p>
    <a href="http://localhost:7001/WebServices/proxy/mazegen/MazeGenerator.jws?.EXPLORE=.TEST">MazeGenerator</a>
    <a href="http://localhost:7001/WebServices/proxy/register/RegisterPerson.jws?.EXPLORE=.TEST">RegisterPerson</a>
  </p>
  <p>Sources: /WebServices/proxy</p>
</td>
</tr>
<tr>
  <td width="25">&nbsp;</td>
  <td width="120" valign="top">&nbsp;</td>
  <td width="5">&nbsp;</td>
  <td valign="top">
    <hr size="1" align="left">
    <p><b>Web service security samples</b></p>

    <table border="0" cellspacing="1" cellpadding="0">
      <tr>
        <td width="5%">&nbsp;</td>
        <td colspan="3" valign="top" style="font-size:10px;">Role-based security samples</td>
      </tr>
      <tr>
        <td colspan="2">&nbsp;</td>
        <td style="font-size:10px;">User Accounts</td>
        <td colspan="2"><a href="http://localhost:7001/WebServices/security/roleBased/createUser">Create User</a>
      </tr>
      <tr>
        <td width="10%" colspan="2">&nbsp;</td>
        <td style="font-size:10px;">Authorization:</td>
        <td colspan="2"><a href="http://localhost:7001/WebServices/security/roleBased/Bank.jws">Bank.jws</a>
      </tr>
      <tr>
        <td width="5%">&nbsp;</td>
        <td colspan="3" valign="top" style="font-size:10px;">Transport security samples</td>
      </tr>
      <tr>
        <td width="10%" colspan="2">&nbsp;</td>
        <td style="font-size:10px;">Basic authentication:</td>
        <td align="left"><a href="http://localhost:7001/WebServices/security/transport/basicAuth">Basic Auth</a>
      </tr>
      <tr>
        <td width="10%" colspan="2">&nbsp;</td>
        <td style="font-size:10px;">Client certificate:</td>
        <td align="left"><a href="http://localhost:7001/WebServices/security/transport/clientCert">Client Cert</a>
      </tr>
      <tr>
        <td width="10%" colspan="2">&nbsp;</td>
        <td style="font-size:10px;">One-way SSL:</td>
        <td align="left"><a href="http://localhost:7001/WebServices/security/transport/helloWorld">Hello World</a>
      </tr>
      <tr>
        <td width="5%">&nbsp;</td>
        <td colspan="3" valign="top" style="font-size:10px;">WS-Security samples</td>
      </tr>
      <tr>
        <td width="10%" colspan="2">&nbsp;</td>
        <td style="font-size:10px;">Callback:</td>
        <td align="left"><a href="http://localhost:7001/WebServices/security/wsse/callback/callBack">Callback</a>
      </tr>
    </table>
  </td>
</tr>

```

navWebServices.html Sample

```
|  |  |  |  |  |  |  | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |  |                                                                                                                                                                                                             | |---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------| | <td colspan="2" width="10%">&amp;nbsp;&lt;/td&gt; <td &gt;&lt;a="" &gt;request="" 10%"="" <td="" align="left" colspan="2" href="http://localhost:7001/WebServices/security/wsse/reqResp/cli &lt;/td&gt;&lt;/tr&gt; &lt;/table&gt; &lt;/td&gt;&lt;/tr&gt; &lt;tr&gt; &lt;td&gt; &lt;table&gt; &lt;tr&gt; &lt;td&gt; &lt;td width=" response:&lt;="" style="font-size:10px;" td&gt;="">&amp;nbsp;&lt;/td&gt; <td &gt;="" &gt;&amp;nbsp;&lt;="" &gt;&lt;a="" &gt;user="" 25"&gt;&amp;nbsp;&lt;="" <hr="" <p="" <td="" align="left" href="http://localhost:7001/WebServices/security/wsse/usertoken/w &lt;/td&gt;&lt;/tr&gt; &lt;/table&gt; &lt;/td&gt;&lt;/tr&gt; &lt;tr&gt; &lt;td&gt; &lt;p&gt;Sources: /WebServices/security&lt;/p&gt; &lt;/td&gt;&lt;/tr&gt; &lt;tr&gt; &lt;td&gt; &lt;table&gt; &lt;tr&gt; &lt;td&gt; &lt;td width=" size="1" style="font-size:10px;" td&gt;="" token:&lt;="" valign="top" width="5">&lt;b&gt;Service control samples&lt;/b&gt;&lt;/p&gt; <p> <a &gt;quotec="" <="" a="" href="http://localhost:7001/WebServices/service/QuoteClient.jws?.EXPLORE=.TEST"> </a></p> <p>Sources: /WebServices/service&lt;/p&gt; </p></td></td></td> | &nbsp;</td> <td &gt;&lt;a="" &gt;request="" 10%"="" <td="" align="left" colspan="2" href="http://localhost:7001/WebServices/security/wsse/reqResp/cli &lt;/td&gt;&lt;/tr&gt; &lt;/table&gt; &lt;/td&gt;&lt;/tr&gt; &lt;tr&gt; &lt;td&gt; &lt;table&gt; &lt;tr&gt; &lt;td&gt; &lt;td width=" response:&lt;="" style="font-size:10px;" td&gt;="">&amp;nbsp;&lt;/td&gt; <td &gt;="" &gt;&amp;nbsp;&lt;="" &gt;&lt;a="" &gt;user="" 25"&gt;&amp;nbsp;&lt;="" <hr="" <p="" <td="" align="left" href="http://localhost:7001/WebServices/security/wsse/usertoken/w &lt;/td&gt;&lt;/tr&gt; &lt;/table&gt; &lt;/td&gt;&lt;/tr&gt; &lt;tr&gt; &lt;td&gt; &lt;p&gt;Sources: /WebServices/security&lt;/p&gt; &lt;/td&gt;&lt;/tr&gt; &lt;tr&gt; &lt;td&gt; &lt;table&gt; &lt;tr&gt; &lt;td&gt; &lt;td width=" size="1" style="font-size:10px;" td&gt;="" token:&lt;="" valign="top" width="5">&lt;b&gt;Service control samples&lt;/b&gt;&lt;/p&gt; <p> <a &gt;quotec="" <="" a="" href="http://localhost:7001/WebServices/service/QuoteClient.jws?.EXPLORE=.TEST"> </a></p> <p>Sources: /WebServices/service&lt;/p&gt; </p></td></td> |  | &nbsp;</td> <td &gt;="" &gt;&amp;nbsp;&lt;="" &gt;&lt;a="" &gt;user="" 25"&gt;&amp;nbsp;&lt;="" <hr="" <p="" <td="" align="left" href="http://localhost:7001/WebServices/security/wsse/usertoken/w &lt;/td&gt;&lt;/tr&gt; &lt;/table&gt; &lt;/td&gt;&lt;/tr&gt; &lt;tr&gt; &lt;td&gt; &lt;p&gt;Sources: /WebServices/security&lt;/p&gt; &lt;/td&gt;&lt;/tr&gt; &lt;tr&gt; &lt;td&gt; &lt;table&gt; &lt;tr&gt; &lt;td&gt; &lt;td width=" size="1" style="font-size:10px;" td&gt;="" token:&lt;="" valign="top" width="5">&lt;b&gt;Service control samples&lt;/b&gt;&lt;/p&gt; <p> <a &gt;quotec="" <="" a="" href="http://localhost:7001/WebServices/service/QuoteClient.jws?.EXPLORE=.TEST"> </a></p> <p>Sources: /WebServices/service&lt;/p&gt; </p></td> |  | <b>Service control samples</b></p> <p> <a &gt;quotec="" <="" a="" href="http://localhost:7001/WebServices/service/QuoteClient.jws?.EXPLORE=.TEST"> </a></p> <p>Sources: /WebServices/service&lt;/p&gt; </p> | |---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------| |
| |                                                                                                                                                                                                                                                                                                                                                                                                                                                             |                                                                                                                                                                                                                                                                                                                         | |-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------| | <td &gt;="" &gt;&amp;nbsp;&lt;="" <hr="" <p="" <td="" align="left" size="1" td&gt;="" valign="top" width="5">&lt;b&gt;Timer control samples&lt;/b&gt;&lt;/p&gt; <p> <a &gt;advanc="" <="" a="" href="http://localhost:7001/WebServices/timer/AdvancedTimer.jws?.EXPLORE=.TEST"> <a &gt;simpleti="" <="" a="" href="http://localhost:7001/WebServices/timer/SimpleTimer.jws?.EXPLORE=.TEST"> </a></a></p> <p>Sources: /WebServices/timer&lt;/p&gt; </p></td> | <b>Timer control samples</b></p> <p> <a &gt;advanc="" <="" a="" href="http://localhost:7001/WebServices/timer/AdvancedTimer.jws?.EXPLORE=.TEST"> <a &gt;simpleti="" <="" a="" href="http://localhost:7001/WebServices/timer/SimpleTimer.jws?.EXPLORE=.TEST"> </a></a></p> <p>Sources: /WebServices/timer&lt;/p&gt; </p> | |-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------| |
| |                                                                                                                                                                                                                                                                                                                                                                                                                 |                                                                                                                                                                                                                                                                             |                                                                                                                                                     |  |  |  | |-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|--|--|--| | <td &gt;="" &gt;&amp;nbsp;&lt;="" <hr="" <p="" <td="" align="left" size="1" td&gt;="" valign="top" width="5">&lt;b&gt;XMLBeans samples&lt;/b&gt;&lt;/p&gt; <p> <table &gt;="" <tr="" border="0" cellpadding="0" cellspacing="1"> <td> <td &gt;&amp;nbsp;&lt;="" &gt;cursor="" <="" <td="" align="top" colspan="3" samples&lt;="" style="font-size:10px;" td="" td&gt;="" width="5%"></td></td></table></p></td> | <b>XMLBeans samples</b></p> <p> <table &gt;="" <tr="" border="0" cellpadding="0" cellspacing="1"> <td> <td &gt;&amp;nbsp;&lt;="" &gt;cursor="" <="" <td="" align="top" colspan="3" samples&lt;="" style="font-size:10px;" td="" td&gt;="" width="5%"></td></td></table></p> | <td &gt;&amp;nbsp;&lt;="" &gt;cursor="" <="" <td="" align="top" colspan="3" samples&lt;="" style="font-size:10px;" td="" td&gt;="" width="5%"></td> |  |  |  | |-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|--|--|--| |
| &nbsp;</td>  Index.jsp Sample | | | | |

```

navWebServices.html Sample

```

        <td width="10%" colspan="2">&nbsp;</td>
        <td colspan="2"><a href="http://localhost:7001/WebServices/xmlBeans/cursor/TokenTypes
</tr>
<tr>
    <td width="5%">&nbsp;</td>
    <td colspan="3" valign="top" style="font-size:10px;">Schema samples</td>
</tr>
<tr>
    <td width="10%" colspan="2">&nbsp;</td>
    <td colspan="2" align="left"><a href="http://localhost:7001/WebServices/xmlBeans/sche
</tr>
<tr>
    <td width="10%" colspan="2">&nbsp;</td>
    <td colspan="2" align="left"><a href="http://localhost:7001/WebServices/xmlBeans/sche
</tr>
<tr>
    <td width="10%" colspan="2">&nbsp;</td>
    <td colspan="2" align="left"><a href="http://localhost:7001/WebServices/xmlBeans/sche
</tr>
<tr>
    <td width="10%" colspan="2">&nbsp;</td>
    <td colspan="2" align="left"><a href="http://localhost:7001/WebServices/xmlBeans/sche
</tr>
<tr>
    <td width="10%" colspan="2">&nbsp;</td>
    <td colspan="2" align="left"><a href="http://localhost:7001/WebServices/xmlBeans/sche
</tr>
<tr>
    <td width="5%">&nbsp;</td>
    <td colspan="3" valign="top" style="font-size:10px;">XQuery samples</td>
</tr>
<tr>
    <td width="10%" colspan="2">&nbsp;</td>
    <td colspan="2" align="left"><a href="http://localhost:7001/WebServices/xmlBeans/xque
</tr>
<tr>
    <td width="10%" colspan="2">&nbsp;</td>
    <td colspan="2" align="left"><a href="http://localhost:7001/WebServices/xmlBeans/xque
</tr>
</table>
</p>
<p>Sources: /WebServices/xmlBeans</p>
</td>
</tr>
<tr>
    <td width="25">&nbsp;</td>
    <td width="120" valign="top">&nbsp;</td>
    <td width="5">&nbsp;</td>
    <td valign="top">
        <hr size="1" align="left">
        <p><b>XQuery Map Samples</b></p>
        <p>
            <a href="http://localhost:7001/WebServices/xqueryMap/InputMapMultiple.jws?.EXPLORE=.TESTXML">Out
            <a href="http://localhost:7001/WebServices/xqueryMap/OutputMap.jws?.EXPLORE=.TESTXML">Out
            <a href="http://localhost:7001/WebServices/xqueryMap/OutputScriptMap.jws?.EXPLORE=.TESTXML">Out
            <a href="http://localhost:7001/WebServices/xqueryMap/SimpleMap.jws?.EXPLORE=.TESTXML">Sim
        </p>
        <p>Sources: /WebServices/xqueryMap</p>
    </td>
</tr>
</tr>

```


navWebServices.html Sample

```
<tr valign="top">
  <td width=25>&nbsp;</td>
  <td width=120 valign="top">&nbsp;</td>
  <td width=5>&nbsp;</td>
  <td valign="top">
    <hr size="1" align="left">
    <p>
      Return to the WebLogic Workshop Samples
      <netui:anchor href="../../GettingStarted/Controller.jspf">Home Page</netui:anchor>
    </p>
    <p>&nbsp;</p>
  </td>
</tr>

</table>
</body>
</html>
```

interop Samples

This section contains source code for the following samples.

Samples Included in This Section

dotNET Samples

dotNET Samples

This section contains source code for the following samples.

Samples Included in This Section

ConversationClient.asmx.cs Sample

Readme.html Sample

ConversationClient.asmx.cs Sample

This topic includes the source code for the ConversationClient.asmx.cs Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/interop/dotNET/

Sample Source Code

```
using System;
using System.ComponentModel;
using System.Diagnostics;
using System.Web.Services;
using System.IO;

namespace test
{
    /// <summary>
    /// This is an example of a .NET web service that is a client of
    /// a WebLogic Workshop web service. The target web service,
    /// samples/async/Conversation.jws, is conversational and optionally
    /// uses callbacks. This client is fully capable of participating in
    /// conversations with Conversation.jws and can also receive callbacks.
    /// </summary>

    /*
     * This Namespace declaration is required to cause the callback
     * handler (onResultReady) to be in the proper namespace. Ideally,
     * we could set the namespace on a per-method basis, but .NET doesn't
     * appear to allow that.
     */
    [WebService(Namespace="http://www.openuri.org/")]
    public class ConversationClient : System.Web.Services.WebService
    {

        public ConversationClient()
        {
            //CODEGEN: This call is required by the ASP.NET Web Services Designer
            InitializeComponent();
        }

        #region Component Designer generated code

        //Required by the Web Services Designer
        private IContainer components = null;

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
        }
    }
}
```

navWebServices.html Sample

```
/// <summary>
/// Clean up any resources being used.
/// </summary>
protected override void Dispose( bool disposing )
{
    if(disposing && components != null)
    {
        components.Dispose();
    }
    base.Dispose(disposing);
}

#endregion

/*
 * start invokes the Conversation web service's startRequest
 * operation.
 *
 * Since the Conversation web service is conversational,
 * we must also prepare a SOAP header containing a conversation ID.
 *
 * Since the Conversation web service can optionally communicate
 * the result of it's work via a callback, we must prepare a
 * second SOAP header containing the "callbackLocation", which is
 * the URL of the recipient to which callbacks should be sent.
 */
[WebMethod(EnableSession=true)]
public void start(Boolean useCallbacks, Boolean useIPAddress)
{
    /*
     * The Conversation proxy was created using .NET's wsdl.exe
     * application and the Conversation.jws's WSDL file. The WSDL
     * file for any WebLogic Workshop web service may be obtained
     * by hitting the web service's URL with "?WSDL" appended to
     * the end. For example:
     *
     * http://somehost:7001/samples/async/Conversation.jws?WSDL
     *
     * wsdl.exe produces a C# proxy class. Place the resulting
     * Conversation.cs file in your .NET project, then use Visual
     * Studio's Project->Add Existing Item menu action to "import"
     * the class into the project.
     */
    Conversation conv;
    String conversationID;
    String callbackLocation;
    int asmIndex;

    /*
     * Construct the callback location from various pieces of
     * server and HttpRequest info.
     */
    Uri requestUrl = Context.Request.Url;

    if( useIPAddress )
    {
        /*
         * if useIPAddress is true, construct the callbackLocation
         * with the IP address of this host.
         */
        callbackLocation = requestUrl.Scheme + "://" +
```

navWebServices.html Sample

```
        System.Net.Dns.GetHostByName(Context.Server.MachineName +
        ":" + requestUrl.Port + requestUrl.AbsolutePath;
    }
    else
    {
        /*
        * if useIPAddress is false, construct the callbackLocation
        * with the hostname of this host.
        */
        callbackLocation = requestUrl.Scheme + "://" +
            Context.Server.MachineName +
            ":" + requestUrl.Port + requestUrl.AbsolutePath;
    }

    // Remove everything after ".asmx"
    asmxIndex = callbackLocation.IndexOf(".asmx") + 5;
    callbackLocation = callbackLocation.Remove(asmxIndex,
        callbackLocation.Length - asmxIndex);

    /*
    * Make up a conversation ID for this conversation. It must
    * be guaranteed unique across all conversations on the target
    * web service's server.
    *
    * Here we use a string composed of the current process ID
    * and the current date and time. That's not completely
    * reliable since this client could possibly start two
    * conversations within the space of a second (the resolution
    * of DateTime.ToString()).
    */
    conversationID = Process.GetCurrentProcess().Id + " - " +
        DateTime.Now.ToString();

    /*
    * Create an instance of the proxy for the Conversation
    * web service.
    */
    conv = new Conversation();

    /*
    * Construct a conversation start header and set conversation ID
    * and callback location.
    */
    conv.StartHeaderValue = new StartHeader();
    conv.StartHeaderValue.conversationID = conversationID;
    conv.StartHeaderValue.callbackLocation = callbackLocation;

    /*
    * Persist the conversationID in session state so that it can
    * be used in other methods that take part in the conversation.
    *
    * This is not safe since one session could start multiple
    * conversations, but there is no other apparent way to persist
    * this information. Member variables of WebService classes
    * are not persisted across method invocations.
    */
    Session["ConversationID"] = conversationID;

    /*
    * Invoke the startRequest method of the web service. The
    * single boolean parameter determines whether the Conversation
```

navWebServices.html Sample

```
* web service will use callbacks to communicate the result
* back to this client.
*
* If the argument is true, an onResultReady callback will
* be sent when the result is ready. This client must implement
* a method with that name that expects the message shape defined
* by the target web service (returns void and accepts a single
* string argument). See the onResultReady method below.
*
* If the argument to startRequest is false, callbacks will not
* be used and this client must use the getRequestStatus method
* to poll the Conversation web service for the result.
*/
conv.startRequest(useCallbacks);
}

/*
 * getStatus invokes Conversation's getRequestStatus method.
 * getRequestStatus is a polling method that is an alternative
 * for web services that cannot receive callbacks.
 *
 * Note that a conversation must be started with startRequest before
 * this method may be invoked. If not, or if this method is invoked
 * outside of a conversation for any reason, it will get back a SOAP
 * fault indicating that the conversation does not exist.
 */
[WebMethod(EnableSession=true)]
public String getStatus()
{
    String result;

    /*
     * Create an instance of the proxy for the Conversation
     * web service. We could probably persist the proxy instance
     * in session state, but chose not to.
     */
    Conversation conv = new Conversation();

    /*
     * Construct a conversation continue header and set the
     * conversation ID to the one we cached on session state in
     * the start method.
     */
    conv.ContinueHeaderValue = new ContinueHeader();
    conv.ContinueHeaderValue.conversationID = (String)Session["ConversationID"];

    /*
     * Invoke the getRequestStatus method of the web service.
     */
    result = conv.getRequestStatus();
    return result;
}

/*
 * finish invokes Conversation's terminateRequest method, which
 * terminates the current conversation.
 *
 * Note that a conversation must be started with startRequest before
 * this method may be invoked. If not, or if this method is invoked
 * outside of a conversation for any reason, it will get back a SOAP
 * fault indicating that the conversation does not exist.
 */
```

navWebServices.html Sample

```
*/
[WebMethod(EnableSession=true)]
public void finish()
{
    /*
     * Create an instance of the proxy for the Conversation
     * web service. We could probably persist the proxy instance
     * in session state, but chose not to.
     */
    Conversation conv = new Conversation();

    /*
     * Construct a conversation continue header and set the
     * conversation ID to the one we cached on session state in
     * the start method. Both "continue" and "finish" methods
     * use the same SOAP header format.
     */
    conv.ContinueHeaderValue = new ContinueHeader();
    conv.ContinueHeaderValue.conversationID = (String)Session["ConversationID"];

    /*
     * Invoke the terminateRequest method of the web service.
     */
    conv.terminateRequest();
}

/*
 * onResultReady is a callback handler for the onResultReady
 * callback that Conversation.jws can optionally use to return
 * its results.
 *
 * .NET does not support callbacks directly, but a callback is just
 * a method invocation message. So if you construct a WebMethod with
 * the same signature as the callback and set the XML namespace
 * properly, it serves as a callback handler.
 *
 * The namespace of this entire WebService is set to match that
 * required by the callback ("http://www.openuri.org/") using the
 * [WebService(Namespace=...)] attribute at the top of this file.
 * Ideally we could set the namespace on a per-method basis, but
 * .NET doesn't appear to allow that.
 */
[WebMethod]
public void onResultReady(String result)
{
    /*
     * When the callback is invoked, log a message to the
     * hardcoded file c:\temp\ConversationClient.log.
     *
     * Note: if c:\temp does not exist on this server, an
     * Exception will be raised. Since it is not handled here,
     * it will be returned as a SOAP fault to the Conversation
     * web service.
     */
    TextWriter output;
    output = File.AppendText("c:\\temp\\ConversationClient.log");
    String msg = "[" + DateTime.Now.ToString() + "] callback received";
    output.WriteLine(msg);
    output.Flush();
    output.Close();
}
```



```
}  
}
```

Readme.html Sample

This topic includes the source code for the Readme.html Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/interop/dotNET/

Sample Source Code

```
<html>
<head>
<title>
WebLogic Workshop&trade; Web Service .NET Client Example
</title>
<style type="text/css">
h1,
h2,
h3,
h4,
p,
li,
blockquote
{
    font-family: Verdana, Arial, Helvetica, sans-serif;
    margin-left: 0px;
}
p, li { font-size: 12; }
blockquote { font-size: 12; }
blockquote { margin-left: 1em; }
ul { list-style-type: square; }
</style>
</head>
<body>
<h1>WebLogic Workshop&trade; Web Service .NET Client Example</h1>
<h2>Introduction</h2>
<p>
This sample demonstrates how to create a .NET client for a WebLogic Workshop web service (a JWS).
The client uses the sample web service Conversation.jws, found in samples/async. Conversation.jws
is a conversational web service that can optionally communicate results to its client via a callback.
The sample .NET client is fully capable of participating in conversations with Conversation.jws
as receiving results via callback.</p>

<h2>Contents</h2>

<blockquote>
    ConversationClient.asmx.cs
</blockquote>
    The C# source file of the ConversationClient .NET web service that acts as a client to Conversation.jws.
    The following code
    below describe how to create a .NET project into which this file will be copied.</p>
</blockquote>
</blockquote>

<h2>Instructions</h2>
```

navWebServices.html Sample

```
<h3>In WebLogic Workshop</h3>
<p>Test Conversation.jws by running it from the WebLogic Workshop visual development
environment or by browsing to <a href="http://localhost:7001/WebServices/async/Conversation.jws">
</a>
</p>
<blockquote>
<p>If you specify "true" for useCallbacks, hit refresh until the <tt>onResultReady</tt>
callback appears.</p>
<p>If you specify "false" for useCallbacks, invoke getRequestStatus until the request
is fulfilled, then invoke terminateRequest.</p>
</blockquote>

<h3>In Visual Studio .NET</h3>
<ol>
<li>Create a new project following these steps:<br><br>
<ol>
<li>Select File->New->Project.</li>
<li>Select "Visual C# Projects".</li>
<li>On the right side, select "ASP.NET Web Service".</li>
<li>Change the location from <tt>http://localhost/WebService1</tt> to <tt>http://localhost/test</tt>
</li>
<li>Select OK.</li>
</ol>
<br>
</li>
<li>Select the Service1.asmx web service in the Solutions Explorer (pane on the right by default)
<li>In the main edit window (which is mostly blank) select the "click here to switch to code view"
<li>Replace the contents of the file (ConversationClient.asmx.cs) with the contents of the same file
<li>Construct a proxy for the Conversation.jws web service by following these steps:<br><br>
<ol>
<li>Get the WSDL file by browsing to the URL of the Conversation with "?WSDL" appended to the URL
<li>Using the browser's "File->Save As...", save the WSDL to a temporary location (e.g. c:\temp)
<li>In a CMD window, cd to the temporary location to which you saved Conversation.wsdl.</li>
<li>Use .NET's wsdl.exe command to create a proxy from the WSDL. Type "wsdl Conversation.wsdl"
<li>Copy the Conversation.cs file to the .NET web service project you created above. If you are in the
<li>Back in Visual Studio .NET, import the proxy class by selecting "Project->Add Existing Item"
</ol>
<br>
</li>
<li>Build the project by pressing Ctrl-Shift-B. If it builds without errors, you are ready to test
</li>
</ol>

<h3>To Run the Client</h3>
<ol>
<li>In Visual Studio .NET, press F5 to run the "solution". A browser will appear with links for
<li>Select "start". When its form appears, enter "true" or "false" (without the quotes) for <tt>useCallbacks</tt>
<li>Invoke the "start" method by clicking the button. If you are watching the log for Conversation.jws,
<li>If you chose to use callbacks, a message will be written to the file c:\temp\ConversationClient.log
<li>If you chose not to use callbacks, you must continually invoke "getStatus" until the status is "complete"
<li>Again, if you have a different browser monitoring the Test View for Conversation.jws, a "request completed"
</li>
</ol>
</html>
```

jms Samples

This section contains source code for the following samples.

Samples Included in This Section

AccountPublish.jws Sample

AccountPublishJMSSControl.jcx Sample

AccountSubscribe.jws Sample

AccountSubscribeJMSSControl.jcx Sample

CustomJMSSClient.jws Sample

CustomJMSSControl.jcx Sample

SimpleJMS.jws Sample

SimpleJMSSControl.jcx Sample

AccountPublish.jws Sample

This topic includes the source code for the AccountPublish.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/jws/

Sample Source Code

```
package jws;

import com.bea.control.JwsContext;
import weblogic.jws.util.Logger;
import org.openuri.bea.samples.workshop.jms.account.AccountTransactionDocument;
import org.openuri.bea.samples.workshop.jms.account.AccountTransactionDocument.AccountTransactionDocument;
import java.math.BigDecimal;

/**
 * <p> Demonstrates using a JMS control to publish to a JMS
 * topic. The companion sample web service AccountSubscribe.jws
 * listens to the same topic, so messages published by this web
 * service will be received by AccountSubscribe.jws (if it
 * is subscribed).</p>
 *
 * <p>Also demonstrates use of built-in logging facilities.
 * The JwsContext.getLogger method will return a log4j
 * logger that may be used by web services to write
 * logging information to (by default) the workshop.log file.</p>
 *
 * <p>Log messages can have several levels:
 * debug, info, warn, error and fatal. Note that a "debug"-level
 * log entry won't be accepted by the log file. By default,
 * logging for this server is configured to accept only
 * those messages of "info" level and higher. You can change
 * this configuration by editing the workshopLogCfg.xml file
 * installed with WebLogic Workshop. See the documentation for more
 * information.</p>
 *
 * <p>Uses the AccountPublishJMSControl.jcx JMS control.</p>
 * @common:target-namespace namespace="http://workshop.bea.com/AccountPublish"
 */
public class AccountPublish implements com.bea.jws.WebService
{
    /**
     * @common:context
     */
    JwsContext context;

    /**
     * @common:control
     */
    private jws.AccountPublishJMSControl accountPublishJMS;
```

navWebServices.html Sample

```
/**
 * <p>Accept transaction information for a deposit from
 * the client and pass it to the JMS control to be
 * published.</p>
 *
 * @common:operation
 */
public void deposit(String accountID, double amount)
{
    /*
     * Get a logger with the name of this web service
     * (that is a convention).
     *
     * This web service is non-conversational, so it
     * cannot persist the logger as a member variable.
     */
    Logger logger = context.getLogger("AccountPublish");

    /*
     * Use the logger to write an "info"-level message
     * to the log.
     */
    logger.info("about to publish (deposit)");

    /*
     * Use the JMS control to publish a message to the
     * topic for which it is configured (in
     * AccountPublishJMSControl.jcx).
     */
    AccountTransactionDocument acctDoc = AccountTransactionDocument.Factory.newInstance();
    AccountTransaction txn = acctDoc.addNewAccountTransaction();
    txn.setTransactionAmount(new BigDecimal(amount));
    accountPublishJMS.deposit(txn,accountID);

    logger.info("done publishing (deposit)");
}

/**
 * <p>Accept transaction information for a withdrawal from
 * the client and pass it to the JMS control to be
 * published.</p>
 *
 * @common:operation
 */
public void withdraw(String accountID, double amount)
{
    /*
     * Get a logger with the name of this web service
     * (that is a convention).
     *
     * This web service is non-conversational, so it
     * cannot persist the logger as a member variable.
     */
    Logger logger = context.getLogger("AccountPublish");

    /*
     * Use the logger to write an "info"-level message
     * to the log.
     */
    logger.info("about to publish (withdraw)");
```

navWebServices.html Sample

```
/*
 * Use the JMS control to publish a message to the
 * topic for which it is configured (in
 * AccountPublishJMSControl.jcx).
 */
AccountTransactionDocument acctDoc = AccountTransactionDocument.Factory.newInstance();
AccountTransaction txn = acctDoc.addNewAccountTransaction();
txn.setTransactionAmount(new BigDecimal(amount));
accountPublishJMS.withdraw(txn, accountID);

logger.info("done publishing (withdraw)");
}

/**
 * The context_onException callback handler can be defined in
 * any web service that declared a JwsContext object (this one
 * happens to declare a JwsContext object named "context").
 *
 * Various errors that occur while executing this web service's
 * methods will be reported by a call to this callback handler.
 */
public void context_onException(Exception ex, String methodName, Object [] args)
{
    /*
     * Use the logger to write the exception's stack trace to
     * the log file.  Exception.printStackTrace wants a
     * java.io.PrintWriter, so make a java.io.StringWriter,
     * wrap it as a PrintWriter and write the exception's
     * stack trace to it.  Then write the StringWriter's
     * string to the log.
     */
    Logger logger = context.getLogger("AccountPublish");
    java.io.StringWriter writer = new java.io.StringWriter();
    ex.printStackTrace(new java.io.PrintWriter(writer));
    logger.info(writer.toString());
}
}
```

AccountPublishJMSControl.jcx Sample

This topic includes the source code for the AccountPublishJMSControl.jcx Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/jms/

Sample Source Code

```
package jms;

import com.bea.control.JMSControl;
import com.bea.control.ControlExtension;
import java.io.Serializable;
import org.openuri.bea.samples.workshop.jms.account.AccountTransactionDocument;
import org.openuri.bea.samples.workshop.jms.account.AccountTransactionDocument.AccountTransacti

/**
 * <p>JMS control used by AccountPublish.jws sample. Defines
 * <b>deposit</b> and <b>withdraw</b> methods that publish
 * appropriately formatted messages to the JMS topic
 * <tt>jms.AccountUpdate</tt>.</p>
 *
 * <p>Notice that this control specifies a value for
 * send-jndi-name but not for receive-jndi-name since
 * this control is only used to publish messages.<p>
 *
 * @jc:jms
 *     send-type="topic"
 *     send-jndi-name="jms.AccountUpdate"
 *     connection-factory-jndi-name="weblogic.jws.jms.QueueConnectionFactory"
 */
public interface AccountPublishJMSControl extends ControlExtension, JMSControl
{
    /**
     * <p><b>deposit</b> encodes the transaction information into
     * message properties (containing the transaction type and
     * account ID) and the message body (containing the transaction
     * amount).</p>
     *
     * @jc:jms-property key="accountIdentifier" value="{accountID}"
     * @jc:jms-property key="transactionType" value="DEPOSIT"
     */
    public void deposit(AccountTransaction transaction, String accountID);

    /**
     * <p><b>withdraw</b> encodes the transaction information into
     * message properties (containing the transaction type and
     * account ID) and the message body (containing the transaction
     * amount).</p>
     *
     * @jc:jms-property key="accountIdentifier" value="{accountID}"
     * @jc:jms-property key="transactionType" value="WITHDRAW"
     */
}
```


navWebServices.html Sample

```
*  
*/  
public void withdraw(AccountTransaction transaction, String accountID);  
}
```

AccountSubscribe.jws Sample

This topic includes the source code for the AccountSubscribe.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/jms/

Sample Source Code

```
package jms;

import com.bea.control.JwsContext;
import weblogic.jws.util.Logger;
import java.text.SimpleDateFormat;
import java.util.Date;
import org.openuri.bea.samples.workshop.jms.account.AccountTransactionDocument;
import org.openuri.bea.samples.workshop.jms.account.AccountTransactionDocument.AccountTransacti
import java.math.BigDecimal;
import java.util.HashMap;
import java.util.Map;

/**
 * <p> Demonstrates using a JMS control to subscribe to a JMS
 * topic. The companion sample web service AccountPublish.jws
 * publishes to the same topic, so messages published by that web
 * service will be received by this one (if it is subscribed).</p>
 *
 * <p>Also demonstrates use of built-in logging facilities.
 * The JwsContext.getLogger method will return a log4j
 * logger that may be used by web services to write
 * logging information to (by default) the workshop.log file.</p>
 *
 * <p>Log messages can have several levels:
 * debug, info, warn, error and fatal. Note that a "debug"-level
 * log entry won't be accepted by the log file. By default,
 * logging for this server is configured to accept only
 * those messages of "info" level and higher. You can change
 * this configuration by editing the workshopLogCfg.xml file
 * installed with WebLogic Workshop. See the documentation for more
 * information.</p>
 *
 * <p>Uses the AccountSubscribeJMSControl.jcx JMS control.</p>
 * @common:target-namespace namespace="http://workshop.bea.com/AccountSubscribe"
 */
public class AccountSubscribe implements com.bea.jws.WebService
{
    /**
     * @common:context
     */
    JwsContext context;

    /**
     * @common:control
```

navWebServices.html Sample

```
*/
private jms.AccountSubscribeJMSControl accountSubscribeJMS;

/*
 * Declare a log4j logger as a member variable. Make it
 * static so it is not persisted. Loggers have no state,
 * so it wouldn't make any sense to persist it.
 */
static private Logger logger;

public Callback callback;

public interface Callback
{
    /**
     * <p>This callback relays the transaction data extracted from the
     * arriving message to the client.</p>
     *
     * @jws:conversation phase="continue"
     */
    public void accountUpdateReceived(
        // JMS Header
        String CorrelationID,
        String Type,
        // JMS Properties
        String accountID,
        String transactionType,
        // Payload
        BigDecimal amount);
}

/**
 * <p>Start listening to the JMS topic. Messages that "arrive"
 * while we are not listening will never be received.</p>
 *
 * @common:operation
 * @jws:conversation phase="start"
 */
public void startListening()
{
    /**
     * Since we are starting a conversation, initialize the
     * logger variable by getting a logger from the JwsContext
     * object.
     */
    logger = context.getLogger("AccountSubscribe");

    /**
     * use the logger to write a "info"-level message
     * to the log. Log messages can have several levels:
     * debug, info, warn, error and fatal.
     */
    logger.info("about to subscribe");

    /**
     * Subscribe to the topic to which
     * AccountSubscribeJMSControl.jcx is configured to listen.
     * No messages will be received by this service unless it
     * is subscribed when the message is published.
     */
    accountSubscribeJMS.subscribe();
}
```

navWebServices.html Sample

```
        logger.info("subscribed");
    }

    /**
     * <p>Stop listening to the JMS topic.</p>
     *
     * @common:operation
     * @jws:conversation phase="finish"
     */
    public void stopListening()
    {
        logger.info("about to unsubscribe");

        /*
         * Unsubscribe from the topic to which
         * AccountSubscribeJMSControl.jcx is configured to listen.
         * No messages will be received after this point unless
         * we subscribe again.
         */
        accountSubscribeJMS.unsubscribe();

        logger.info("unsubscribed");
    }

    /**
     * <p>Handler for the receiveUpdate callback of the JMS control.
     * Just calls the client callback to relay the (slightly reordered)
     * data extracted from the message.</p>
     */
    private void accountSubscribeJMS_receiveUpdate(
                                                //Message
                                                AccountTransaction transaction,
                                                // JMS Headers
                                                String CorrelationID,
                                                String DeliveryMode,
                                                String Expiration,
                                                String MessageID,
                                                String Priority,
                                                String Redelivered,
                                                long Timestamp,
                                                String Type,
                                                // Properties
                                                String transactionType,
                                                String accountID)
    {
        logger.info("received message, calling accountUpdateReceived callback");

        /*
         * Call the client callback, passing the data received from the JMS control.
         *
         * The Timestamp is converted from its native long format to a String format
         * using a java.text.SimpleDateFormat object.
         */

        callback.accountUpdateReceived(
                                    CorrelationID,
                                    Type,
                                    accountID,
                                    transactionType,
                                    transaction.getTransactionAmount());
    }
}
```

navWebServices.html Sample

```
        logger.info("called accountUpdateReceived callback");
        if(accountID.equalsIgnoreCase("FINISH")) {
            logger.info("calling fininsh");
            accountSubscribeJMS.unsubscribe();
            context.finishConversation();
        }
    }
}
/**
 * The context_onException callback handler can be defined in
 * any web service that declared a JwsContext object (this one
 * happens to declare a JwsContext object named "context").
 *
 * Various errors that occur while executing this web service's
 * methods will be reported by a call to this callback handler.
 */
public void context_onException(Exception ex, String methodName, Object [] args)
{
    /*
     * Use the logger to write the exception's stack trace to
     * the log file.  Exception.printStackTrace wants a
     * java.io.PrintWriter, so make a java.io.StringWriter,
     * wrap it as a PrintWriter and write the exception's
     * stack trace to it.  Then write the StringWriter's
     * string to the log.
     */
    java.io.StringWriter writer = new java.io.StringWriter();
    ex.printStackTrace(new java.io.PrintWriter(writer));
    logger.info(writer.toString());
}
}
```

AccountSubscribeJMSControl.jcx Sample

This topic includes the source code for the AccountSubscribeJMSControl.jcx Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/jms/

Sample Source Code

```
package jms;

import com.bea.control.JMSControl;
import java.io.Serializable;

import org.openuri.bea.samples.workshop.jms.account.AccountTransactionDocument;
import org.openuri.bea.samples.workshop.jms.account.AccountTransactionDocument.AccountTransactionDocumentType;

/**
 *
 * <p>JMS control used by AccountSubscribe.jws sample. Defines
 * a single callback named <b>receiveUpdate</b> that is called
 * when a message arrives on the <tt>jms.AccountUpdate</tt>
 * topic (while subscribed).</p>
 *
 * <p>Note that in all messaging scenarios the sender and
 * receiver must agree on the message format.</p>
 *
 * <p>Notice that this control specifies a value for
 * receive-jndi-name but not for send-jndi-name since
 * this control is only used to subscribe to messages.<p>
 *
 * @jc:jms receive-type="topic" receive-jndi-name="jms.AccountUpdate"
 * connection-factory-jndi-name="weblogic.jws.jms.QueueConnectionFactory"
 */
public interface AccountSubscribeJMSControl extends com.bea.control.ControlExtension, JMSControl
{
    interface Callback extends JMSControl.Callback
    {
        /**
         * <p>Callback that will be invoked whenever a message arrives
         * on the topic of interest.</p>
         *
         * <p>JMS controls that specify a receive-jndi-name may define
         * exactly one callback that will be invoked when a message
         * arrives. </p>
         *
         * <p>Messages arriving on this topic are expected to have the
         * transaction type encoded in a JMS property named transactionType,
         * the account ID encoded in a JMS property named accountIdentifier,
         * and the transaction amount in the message body.</p>
         *
         * <p>JMS messages may also have JMS message headers present. Which
         * headers (if any) that are set on arriving messages is determined by

```

navWebServices.html Sample

```
* the JMS server.</p>
*
* <p>Theheaders that may be present are:<p>
* <ul><li>
* JMSCorrelationID
* </li><li>
* JMSDeliveryMode
* </li><li>
* JMSEExpiration
* </li><li>
* JMSMessageID
* </li><li>
* JMSPriority
* </li><li>
* JMSRedelivered
* </li><li>
* JMSTimestamp
* </li><li>
* JMSType
* </li></ul>
*
* @jc:jms-headers
*   JMSCorrelationID="{CorrelationID}"
*   JMSDeliveryMode="{DeliveryMode}"
*   JMSEExpiration="{Expiration}"
*   JMSMessageID="{MessageID}"
*   JMSPriority="{Priority}"
*   JMSRedelivered="{Redelivered}"
*   JMSTimestamp="{Timestamp}"
*   JMSType="{Type}"
*
* @jc:jms-property key="accountIdentifier" value="{accountID}"
* @jc:jms-property key="transactionType" value="{transactionType}"
*
*/
public void receiveUpdate(
    // Message Body
    AccountTransaction transaction,

    // Headers
    String CorrelationID,
    String DeliveryMode,
    String Expiration,
    String MessageID,
    String Priority,
    String Redelivered,
    long Timestamp,
    String Type,

    // Properties
    String transactionType,
    String accountID
);
}
```

CustomJMSClient.jws Sample

This topic includes the source code for the CustomJMSClient.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/jms/

Sample Source Code

```
package jms;

/**
 * <p>This web service shows the use of a customized jms control.</p>
 *
 * <p>It uses the same queue for send and receive, so it will send messages to
 * itself (defined in CustomJMSControl). The queue used by this sample is
 * preconfigured in the cg domain when WebLogic Server is installed.</p>
 *
 * <p><i>sendPerson</i>" will publish a message to the send queue.</p>
 *
 * <p><i>onPersonJMS</i>" will receive the message from the receive queue
 * (in this case, the message just sent from publish).</p>
 * @common:target-namespace namespace="http://workshop.bea.com/CustomJMSClient"
 */
public class CustomJMSClient implements com.bea.jws.WebService
{
    /**
     * @common:control
     */
    private jms.CustomJMSControl myCustomQ;

    public Callback callback;

    public interface Callback {
        /**
         * @jws:conversation phase="finish"
         */
        public void onResponse(String msg);
    }

    /**
     * @common:operation
     * @jws:conversation phase="start"
     */
    public void sendPerson(String firstname, String lastname) throws Exception {

        String personname = firstname + " " + lastname;
        myCustomQ.sendPersonJMS( personname,
                                "red property", "blue property");
    }

    public void myCustomQ_onPersonJMS(String personname,
                                      String prop1, String prop2) {
```


navWebServices.html Sample

```
callback.onResponse(  
    "Hello, " + personname +  
    ". You supplied property values '" + prop1 + "' and '" + prop2 + "'." );  
}  
}
```

CustomJMSControl.jcx Sample

This topic includes the source code for the CustomJMSControl.jcx Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/jms/

Sample Source Code

```
package jms;

import com.bea.control.JMSControl;
import java.io.Serializable;

/**
 * <p>This customized JMS control demonstrates the use of method annotations for
 * using XML based JMS messages.</p>
 *
 * <p>It uses the same queue for send and receive, so it will send messages to
 * itself.</p>
 *
 * <p>"<i>sendPersonJMS</i>" will publish a message to the send queue with
 * property values specified.</p>
 *
 * <p>"<i>onPersonJMS</i>" defines a callback that will extract values from both the
 * message and JMS properties.</p>
 */

/**
 * @jc:jms
 *   send-type="queue"
 *   send-jndi-name="jms.CustomJmsCtlQ"
 *   receive-type="queue"
 *   receive-jndi-name="jms.CustomJmsCtlQ"
 *   connection-factory-jndi-name="weblogic.jws.jms.QueueConnectionFactory"
 */
public interface CustomJMSControl extends com.bea.control.ControlExtension, JMSControl
{
    /**
     * @jc:jms-property key="prop1" value="{p1}"
     * @jc:jms-property key="prop2" value="{p2}"
     */
    public void sendPersonJMS(String personname,
                             String p1,
                             String p2);

    public static interface Callback extends JMSControl.Callback
    {
        /**
         * <p>the "<i>onJMSMessage</i>" event is generated whenever a message arrives.</p>
         *
         * @jc:jms-property key="prop1" value="{p1}"
         */
    }
}
```

navWebServices.html Sample

```
* @jc:jms-property key="prop2" value="{p2}"
*
*/
public void onPersonJMS(String pn,
                        String p1,
                        String p2) throws Exception;
    }
}
```

SimpleJMS.jws Sample

This topic includes the source code for the SimpleJMS.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/jms/

Sample Source Code

```
package jms;

/**
 * <p>This web service shows the use of a simple jms control.</p>
 *
 * <p>It uses the same queue for send and receive, so it will send messages to
 * itself. The queue used by this sample is preconfigured in the cg domain
 * when WebLogic Server is installed.</p>
 *
 * <p>"sendString" will publish a text message to the send queue.</p>
 *
 * <p>"onTextMessage" will receive the message from the receive queue
 * (in this case, the message just sent from publish). </p>
 * @common:target-namespace namespace="http://workshop.bea.com/SimpleJMS"
 */
public class SimpleJMS implements com.bea.jws.WebService
{
    /**
     * @common:control
     */
    private jms.SimpleJMSControl myJMSControl;

    private String name;

    public Callback callback;

    public interface Callback {
        public void onMessageReceived(String msg);
    }

    /**
     * <p>Starts an instance of the service and assigns a name to it.
     *
     * @common:operation
     * @jws:conversation phase="start"
     */
    public void start(String name) {
        this.name = name;
    }

    /**
     * <p>Finishes this conversation instance.
     *
     * @common:operation
     * @jws:conversation phase="finish"
     */
}
```

navWebServices.html Sample

```
    */
    public void finish() {
    }

    /**
     * <p>Submits a string message that is sent to the send queue. In this example
     * it will immediately arrive on the receive queue.</p>
     *
     * @common:operation
     * @jws:conversation phase="continue"
     */
    public void sendString(String msg) throws Exception {
        myJMSControl.publishText(msg);
    }

    public void myJMSControl_onTextMessage(String msg) {
        callback.onMessageReceived "[" + name + "] Your message is: " + msg );
    }
}
```

SimpleJMSControl.jcx Sample

This topic includes the source code for the SimpleJMSControl.jcx Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/jms/

Sample Source Code

```
package jms;

import com.bea.control.JMSControl;
import java.io.Serializable;

/**
 * <p>The JMS control that this sample demonstrates. Notice that the
 * send and receive queue are the same.</p>
 *
 * @jc:jms send-type="queue" send-jndi-name="jms.SimpleJmsQ" receive-type="queue" receive-jndi-
 * connection-factory-jndi-name="weblogic.jws.jms.QueueConnectionFactory"
 */
public interface SimpleJMSControl extends com.bea.control.ControlExtension, JMSControl
{
    public void publishText(String msg);

    public static interface Callback extends JMSControl.Callback
    {
        public void onTextMessage(String msg);
    }
}
```

jms_xmlProtocol Samples

This section contains source code for the following samples.

Samples Included in This Section

ClientJWS.jws Sample

ClientXML.java Sample

JMS_XMLProtocol.jws Sample

ClientJWS.jws Sample

This topic includes the source code for the ClientJWS.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/jms_xmlProtocol/

Sample Source Code

```
package jms_xmlProtocol;

import jms_xmlProtocol.ClientXML;

public class ClientJWS implements com.bea.jws.WebService
{
    static final long serialVersionUID = 1L;

    /**
     * Enter a string value between the <name> and </name> tags.
     * <p>This value is passed as a parameter to the ClientXML.java class.
     * <p>ClientXML.java, in turn, constructs a SOAP message to invoke JMS_XMLProtocol.jws.
     * See the server console window for the response.
     *
     * @common:operation
     * @jws:protocol http-xml="true" form-get="false" form-post="false"
     */
    public void requestHello(String name) throws Exception
    {
        ClientXML client = new ClientXML();
        String [] args ={"t3://localhost:7001", name};
        client.main(args);
    }
}
```


ClientXML.java Sample

This topic includes the source code for the ClientXML.java Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/jms_xmlProtocol/

Sample Source Code

```
package jms_xmlProtocol;

import java.io.IOException;
import java.util.Hashtable;
import javax.jms.*;
import javax.naming.Context;
import javax.naming.InitialContext;
import javax.naming.NamingException;

/**
 * This example shows how to invoke a web service via a JMS queue.
 */
public class ClientXML
{
    // Defines the JNDI context factory.
    public final static String JNDI_FACTORY="weblogic.jndi.WLInitialContextFactory";

    // Defines the JMS connection factory.
    public final static String JMS_FACTORY="weblogic.jws.jms.QueueConnectionFactory";

    // Defines the queue.
    public final static String QUEUE="jws.queue";

    private QueueConnectionFactory qconFactory;
    private QueueConnection qcon;
    private QueueSession qsession;
    private QueueSender qsender;
    private Queue queue;
    private TextMessage msg;

    /**
     * Creates all the necessary objects for sending
     * messages to a JMS queue.
     */
    public void init(Context ctx, String queueName)
        throws NamingException, JMSEException
    {
        qconFactory = (QueueConnectionFactory) ctx.lookup(JMS_FACTORY);
        qcon = qconFactory.createQueueConnection();
        qsession = qcon.createQueueSession(false, Session.AUTO_ACKNOWLEDGE);
        queue = (Queue) ctx.lookup(queueName);
        qsender = qsession.createSender(queue);
        msg = qsession.createTextMessage();
        msg.setStringProperty("URI", "/WebServices/jms_xmlProtocol/JMS_XMLProtocol.jws");
    }
}
```

navWebServices.html Sample

```
    qcon.start();
}

/**
 * Sends a message to a JMS queue.
 */
public void send(String message) throws JMSEException {
    msg.setText(message);
    qsender.send(msg);
}

/**
 * Closes the JMS objects.
 */
public void close() throws JMSEException {
    qsender.close();
    qsession.close();
    qcon.close();
}

public void main(String[] args) throws Exception
{
    InitialContext ic = getInitialContext(args[0]);
    ClientXML qs = new ClientXML();
    qs.init(ic, QUEUE);
    readAndSend(qs, args[1]);
    qs.close();
}

private static void readAndSend(ClientXML qs, String name)
    throws IOException, JMSEException
{
    String line = null;
    boolean quitNow = false;
    do
    {
        line="<string xmlns=\"http://www.openuri.org/\">" + name + "</string>";
        if (line != null && line.trim().length() != 0)
        {
            qs.send(line);
            System.out.println("JMS Message Sent: "+line+"\n");
            quitNow = true;
        }
    } while (! quitNow);
}

private static InitialContext getInitialContext(String url)
    throws NamingException
{
    Hashtable env = new Hashtable();
    env.put(Context.INITIAL_CONTEXT_FACTORY, JNDI_FACTORY);
    env.put(Context.PROVIDER_URL, url);
    return new InitialContext(env);
}
}
```

JMS_XMLProtocol.jws Sample

This topic includes the source code for the JMS_XMLProtocol.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/jms_xmlProtocol/

Sample Source Code

```
package jms_xmlProtocol;

/**
 * @jws:protocol jms-xml="true" http-soap="false" form-get="false" form-post="false"
 */
public class JMS_XMLProtocol implements com.bea.jws.WebService
{
    static final long serialVersionUID = 1L;

    /**
     * @common:operation
     * @jws:protocol jms-xml="true" http-xml="false" http-soap="false" form-get="false" form-post="false"
     */
    public void hello(String msg) throws Exception
    {
        System.out.println( "Hello, " + msg);
    }
}
```

localControls Samples

This section contains source code for the following samples.

Samples Included in This Section

asynch Samples

basics Samples

LocalControlTest.jws Sample

nestedControls Samples

tags Samples

asynch Samples

This section contains source code for the following samples.

Samples Included in This Section

BufferJC.java Sample

BufferJCImpI.jcs Sample

BufferJC.java Sample

This topic includes the source code for the BufferJC.java Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/localControls/asynch/

Sample Source Code

```
package localControls.asynch;

import com.bea.control.Control;

/**
 * Represents the public interface for the BufferJC control.
 * BufferJCImpl.jcs implements this interface, providing
 * the control's runtime logic.
 */
public interface BufferJC extends Control
{
    interface Callback
    {
        void onResultsReady(String results);
    }

    /**
     * The startRequest method starts a timer. When the timer's
     * onTimeout callback executes, this control's onResultsReady
     * callback executes, notifying the client that the results are ready.
     *
     * @common:operation
     * @common:message-buffer enable="true"
     */
    void startRequest();
}
```

BufferJCSmpl.jcs Sample

This topic includes the source code for the BufferJCSmpl.jcs Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/localControls/asynch/

Sample Source Code

```
package localControls.asynch;

/**
 * This Java control illustrates how you can add message buffering to
 * control operations to handle heavy loads. With a buffered method,
 * requests to the method are queued and processed as the control has
 * resources available.
 *
 * @jcs:jc-jar
 *   label="Buffered Java Control"
 *   palette-priority="3"
 *   group-name="SampleControls"
 *   version="1.0"
 *   description="A control whose methods are buffered for asynchronous use."
 *   @editor-info:code-gen control-interface="true"
 */
public class BufferJCSmpl implements BufferJC, com.bea.control.ControlSource
{
    /**
     * @common:control
     * @jc:timer timeout="5 seconds"
     */
    private com.bea.control.TimerControl delayTimer;

    public Callback callback;

    /**
     * The startRequest method starts a timer. When the timer's
     * onTimeout callback executes, this control's onResultsReady
     * callback executes, notifying the client that the results
     * are ready.
     *
     * @common:operation
     * @common:message-buffer enable="true"
     */
    public void startRequest()
    {
        delayTimer.start();
    }

    /**
     * This onTimeout callback handler receive the delayTimer control's
     * callback five seconds after the control's start method was called.
     * This code invokes the BufferJC control's onResultsReady callback,
     * which is in turn received by its client.
     */
}
```

navWebServices.html Sample

```
    */  
    public void delayTimer_onTimeout(long arg0)  
    {  
        delayTimer.stop();  
        callback.onResultsReady("Results are ready!");  
    }  
}
```


basics Samples

This section contains source code for the following samples.

Samples Included in This Section

Hello.java Sample

HelloImpl.jcs Sample

Hello.java Sample

This topic includes the source code for the Hello.java Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/localControls/basics/

Sample Source Code

```
package localControls.basics;

import com.bea.control.Control;

/**
 * The public interface for the Hello Java control. A control
 * interface such as this one lists the methods and callbacks that
 * a Java control exposes. The Hello control has just one method,
 * sayHello, that returns a greeting as a String.
 *
 * This controls implementation -- the code that specifies what it
 * does -- is contained in HelloImpl.jcs.
 */
public interface Hello extends Control
{
    /**
     * Test the Hello control using the LocalControlTest web service.
     */
    java.lang.String sayHello();
}
```

HelloImpl.jcs Sample

This topic includes the source code for the HelloImpl.jcs Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/localControls/basics/

Sample Source Code

```
package localControls.basics;

/**
 * This control offers a simple "Hello, World!" response. It illustrates
 * a Java control at its most basic.
 *
 * Java controls you create, like this one, will include at least
 * two files that contain source code. One, a JCS file such as this
 * one, contains the control's implementation code; this defines what
 * the control's methods do. The other file, a control interface file such
 * as Hello.java, contains the control's public interface; it defines
 * the list of methods and callbacks the control exposes. The control
 * implementation file "implements" the interface, putting meaningful
 * logic behind the interface.
 *
 * When you are designing and building a Java control, you will likely
 * never need to manually edit the interface file. If you keep the
 * code-gen annotation's control-interface attribute (below) set to
 * "true", WebLogic Workshop will keep the interface in sync with the
 * implementation as you edit the JCS file.
 *
 * Another interface a Java control implements, ControlSource, enables
 * the control to be serializable. This is essential for a class
 * that contains data that should be preserved throughout a
 * conversation, and allows a Java control to be used by conversational
 * web services.
 *
 * Other annotations at the top of the source code specify the
 * group under which the control should appear in WebLogic
 * Workshop menus, the description that should appear in the
 * Property Editor, its name in the IDE, and so on.
 *
 * @jcs:jcs-jar
 *   label="Hello Control"
 *   palette-priority="3"
 *   group-name="SampleControls"
 *   version="1.0"
 *   description='A simple Java control with a "Hello World" message.'
 *   @editor-info:code-gen control-interface="true"
 */
public class HelloImpl implements Hello, com.bea.control.ControlSource
{
    /**
     * Test the Hello control using the LocalControlTest web service.
     */
}
```

navWebServices.html Sample

```
* @common:operation
*/
public String sayHello()
{
    return "Hello, World!";
}
}
```

LocalControlTest.jws Sample

This topic includes the source code for the LocalControlTest.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/localControls/

Sample Source Code

```
package localControls;

/**
 * @common:target-namespace namespace="http://workshop.bea.com/LocalControlTest"
 */
public class LocalControlTest implements com.bea.jws.WebService
{
    /**
     * @common:control
     * @jc:databaseActions checkInventory="true"
     */
    private localControls.tags.POVerify poVerifyControl;

    /**
     * @common:control
     */
    private localControls.nestedControls.VerifyFunds verifyFundsControl;

    /**
     * @common:control
     */
    private localControls.basics.Hello helloControl;

    /**
     * @common:control
     */
    private localControls.asynch.BufferJC bufferedControl;

    public Callback callback;

    /**
     * Test the Hello control by clicking the testHello button. Test View
     * will refresh to display the control's response, a "Hello, World!"
     * greeting.
     *
     * @common:operation
     */
    public String testHello()
    {
        return helloControl.sayHello();
    }

    /**
     * Test the BufferJC control by clicking the testBuffered button in Test View,
     * then click Refresh until the control's response displays as the

```

navWebServices.html Sample

```
* results of its callback.
*
* The BufferJC control demonstrates how a control
* can support message buffering. The BufferJC control's startRequest
* method is buffered, which means that requests to the method are
* automatically queued and fed to the control as resources allow. <br/><br/>
*
* The BufferJC control is also asynchronous, using a callback to
* send its response to its client (this web service). <br/><br/>
*
* @common:operation
* @jws:conversation phase="start"
*/
public void testBuffered()
{
    bufferedControl.startRequest();
}

/**
 * This callback handler receives the BufferJC control's onResultsReady
 * callback, then forwards the results to this web service's
 * onBufferTestDone callback.
 */
public void bufferedControl_onResultsReady(java.lang.String bufferResults)
{
    callback.onBufferTestDone(bufferResults);
}

public interface Callback
{
    /**
     * Sends a response to the client of this web service when the BufferJC
     * control's callback is received.
     *
     * @jws:conversation phase="finish"
     */
    void onBufferTestDone(String results);

    /**
     * Sends a response to the client of this web service when the VerifyFunds
     * control's callback is received.
     *
     * @jws:conversation phase="finish"
     */
    void onVerifyFundsTestDone(String responseMessage);
}

/**
 * Tests the VerifyFunds control, which submits purchase
 * order information after ensuring that inventory and
 * the customer's account support it. <br/><br/>
 *
 * To test the control, enter a PO number (a combination of numerals
 * that results in a legal int value; letters are allowed, and will be
 * removed by the POUtil control), customer ID (numerals only),
 * item number, and item quantity. The following table lists
 * item numbers and quantities known to the database when
 * WebLogic Workshop is installed. The VerifyFunds control
 * decreases quantities with successful orders. This web service
 * gives the customer a balance of $200.00 to spend for each
```

navWebServices.html Sample

```

* request. You can change this amount to test with another.<br/>
*
* <table style="font-size: 8 pt">
* <tr><td style="width: 100px">itemNumber</td><td style="width: 100px">quantity</td><td>pr
*
* <tr><td>624</td><td>5</td><td>34.95</td></tr>
* <tr><td>625</td><td>6</td><td>39.95</td></tr>
* <tr><td>629</td><td>10</td><td>34.95</td></tr>
* <tr><td>631</td><td>15</td><td>65.95</td></tr>
* <tr><td>640</td><td>1</td><td>24.95</td></tr>
*
* </table>
*
* @param poNumber A number for this purchase order. May be any
* string whose concatenated numerals result in a legal int; non-numeric
* characters are allowed, and will be removed by the control.
* @param customerID The customer's unique ID. This may be any number.
* @param item The number for the item being ordered. Use a number from
* the range given above.
* @param quantity The number of the specified items to order. Note that
* this number must be lower than the quantities given above in order
* for this order to succeed. Give larger numbers to test the control's
* inventory checking. Also, keep in mind that a successful PO transaction
* will decrease the inventory quantity in the database by the amount
* ordered.
*
* @common:operation
* @jws:conversation phase="start"
* @jws:message-buffer enable="true"
*/
public void testVerifyFunds(String poNumber,
    String customerID, int item, int quantity)
{
    /*
    * The amount the customer can spend. Changing this amount may
    * change the results of the control's check to verify customer
    * funds for the purchase.
    */
    double balance = 200.00;

    /* Call the VerifyFunds control with the details of the purchase order. */
    verifyFundsControl.submitPO(poNumber, customerID, item, quantity, balance);
}

/**
* A handler for the VerifyFunds control's onTransactionComplete callback.
* The results are essentially summarized for a callback of this web service.
*/
public void verifyFundsControl_onTransactionComplete(java.lang.String message,
    boolean isBalanceAvailable, boolean isInventoryAvailable)
{
    callback.onVerifyFundsTestDone(message +
        "\n Balance available: " + isBalanceAvailable +
        "\n Inventory available: " + isInventoryAvailable);
}

/**
* Tests the POverify control, which submits purchase order information,
* and will first check inventory if its checkInventory attribute
* is set to "true" (as it is by default).<br/><br/>
*

```

navWebServices.html Sample

```
* To test the control, enter any PO number (a combination of numerals
* that results in a legal int value; dashes are allowed, and will be
* removed by the POVerify control) and customer ID (numerals only), but
* use the item numbers and values given in the following table
* to test the control's inventory checking functionality. <br/><br/>
*
* <table style="font-size: 8 pt">
* <tr><td style="width: 100px">itemNumber</td><td style="width: 100px">quantity</td><td>pr</td></tr>
* <tr><td>624</td><td>5</td><td>34.95</td></tr>
* <tr><td>625</td><td>6</td><td>39.95</td></tr>
* <tr><td>629</td><td>10</td><td>34.95</td></tr>
* <tr><td>631</td><td>15</td><td>65.95</td></tr>
* <tr><td>640</td><td>1</td><td>24.95</td></tr>
* </table>
* <br/>
*
* After entering the parameter values in Test View, then click the
* testPOVerify button. After Test View refreshes, scroll to the bottom
* to see whether the response is "true" (the purchase order addition
* succeeded) or "false" (because there isn't enough inventory to support
* the purchase).<br/><br/>
*
* By default, the control's checkInventory property value is set to "true".
* To set it to "false", use the following steps:
*
* <ol>
* <li>Switch to Design View for the web service.</li>
* <li>On the right side of the design, click the poVerifyControl.</li>
* <li>In the Property Editor, under databaseActions, set the checkInventory
* property value to "false".</li>
* </ol>
*
* With the value set to "false", inventory will no longer be checked and
* the web service will return "true" regardless of whether there are
* enough items.
*
* @param poNumber A number for this purchase order. May be any
* number that is a legal int; dashes will be removed.
* @param customerID The customer's unique ID. This may be any number.
* @param itemNum The number for the item being ordered. Use a number from
* the range given above.
* @param itemQuant The number of the specified items to order. Note that
* this number must be lower than the quantities given above in order
* for this order to succeed. Give larger numbers to test the control's
* inventory checking. Also, keep in mind that a successful PO transaction
* will decrease the inventory quantity in the database by the amount
* ordered.
*
* @common:operation
*/
public boolean testPOVerify(String poNumber, int customerID,
    int itemNum, int itemQuant)
{
    return poVerifyControl.addPurchaseOrder(poNumber, customerID,
        itemNum, itemQuant);
}
```


nestedControls Samples

This section contains source code for the following samples.

Samples Included in This Section

CustomerAccountEJB.jcx Sample

ItemsDatabase.jcx Sample

poUtil Samples

VerifyFunds.java Sample

VerifyFundsImpl.jcs Sample

CustomerAccountEJB.jcx Sample

This topic includes the source code for the CustomerAccountEJB.jcx Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/localControls/nestedControls/

Sample Source Code

```
package localControls.nestedControls;

import com.bea.control.*;

/**
 * An EJB control to support the VerifyFunds sample control. Represents
 * a customer account EJB in purchase order requests.
 *
 * @jc:ejb home-jndi-name="ejb20-containerManaged-AccountHome"
 * @editor-info:ejb home="AccountEJB.jar" bean="AccountEJB.jar"
 */
public interface CustomerAccountEJB
    extends com.bea.control.ControlExtension,
           com.bea.control.EntityEJBControl,
           examples.ejb20.basic.containerManaged.Account,
           examples.ejb20.basic.containerManaged.AccountHome
{ }
```

ItemsDatabase.jcx Sample

This topic includes the source code for the ItemsDatabase.jcx Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/localControls/nestedControls/

Sample Source Code

```
package localControls.nestedControls;

import com.bea.control.*;
import java.sql.SQLException;

/**
 * A Database control to support the VerifyFunds sample control. Provides access
 * to a database for purchase order requests.
 *
 * @jc:connection data-source-jndi-name="cgSampleDataSource"
 */
public interface ItemsDatabase
    extends DatabaseControl, com.bea.control.ControlExtension
{
    /**
     * Select item price based on item number.
     *
     * @jc:sql statement="SELECT price FROM items WHERE itemnumber = {itemNumber}"
     */
    double selectItemPrice(int itemNumber);

    /**
     * Insert purchase and customer information into a table that correlates the two.
     *
     * @jc:sql statement="INSERT INTO po_customers (orderid, customerid) VALUES ({poNumber}, {c"
     */
    void insertItemCustomer(int poNumber, int customerID);

    /**
     * Insert purchase order and item information into a table that correlates the two.
     *
     * @jc:sql statement="INSERT INTO po_items (orderid, itemnumber, quantity) VALUES ({poNumber"
     */
    void insertPOItem(int poNumber, int itemNumber, int quantity);

    /**
     * Select the number of items available based on item number.
     *
     * @jc:sql statement="SELECT quantityAvailable FROM items WHERE itemNumber = {itemNumber}"
     */
    int checkInventory(int itemNumber);

    /**
     * Update the item inventory.
     *
     */
}
```

navWebServices.html Sample

```
* @jc:sql statement="UPDATE items SET quantityAvailable = {newQuantityAvailable} WHERE ite
*/
int updateInventory(int itemNumber, int newQuantityAvailable);
}
```

poUtil Samples

This section contains source code for the following samples.

Samples Included in This Section

POUtil.java Sample

POUtilImpl.jcs Sample

POUtil.java Sample

This topic includes the source code for the POUtil.java Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/localControls/nestedControls/poUtil/

Sample Source Code

```
package localControls.nestedControls.poUtil;

import com.bea.control.Control;

public interface POUtil extends Control
{
    /**
     * Formats a purchase order number, removing non-numeric characters.
     */
    int formatNumber(java.lang.String stringNumber);
}
```

POUtilImpl.jcs Sample

This topic includes the source code for the POUtilImpl.jcs Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/localControls/nestedControls/poUtil/

Sample Source Code

```
package localControls.nestedControls.poUtil;

import java.util.StringTokenizer;

/**
 * Provides a number formatting utility function for the
 * VerifyFunds sample control.
 *
 * @jcs:jcs-jar label="POUtil"
 * @editor-info:code-gen control-interface="true"
 */
public class POUtilImpl implements POUtil, com.bea.control.ControlSource
{
    /**
     * Formats a purchase order number, removing
     * non-numeric characters.
     *
     * @common:operation
     */
    public int formatNumber(String stringNumber)
    {
        StringBuffer delimiters = new StringBuffer();

        for (int i = 0; i < stringNumber.length(); i++)
        {
            if (stringNumber.charAt(i) < '0' || stringNumber.charAt(i) > '9')
                delimiters.append(stringNumber.charAt(i));
        }

        StringTokenizer tokens = new StringTokenizer(stringNumber, delimiters.toString());
        StringBuffer cleanString = new StringBuffer();

        while(tokens.hasMoreTokens())
        {
            cleanString.append(tokens.nextToken());
        }
        int number = new Integer(cleanString.toString()).intValue();
        return number;
    }
}
```

VerifyFunds.java Sample

This topic includes the source code for the VerifyFunds.java Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/localControls/nestedControls/

Sample Source Code

```
package localControls.nestedControls;

import com.bea.control.Control;

/**
 * This is the public interface for the VerifyFunds control.
 * The control is implemented in VerifyFundsImpl.jcs.
 */
public interface VerifyFunds extends Control
{
    /**
     * A Callback interface to support sending messages back to the client.
     */
    interface Callback
    {
        void onTransactionComplete(String message, boolean isBalanceAvailable,
                                   boolean isInventoryAvailable);
    }

    /**
     * @common:operation
     */
    void submitPO(java.lang.String poNumberString, java.lang.String customerIDString, int itemN
}
```


VerifyFundsImpl.jcs Sample

This topic includes the source code for the VerifyFundsImpl.jcs Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/localControls/nestedControls/

Sample Source Code

```
package localControls.nestedControls;

import com.bea.control.ControlException;
import examples.ejb20.basic.containerManaged.Account;
import examples.ejb20.basic.containerManaged.ProcessingErrorException;
import java.rmi.RemoteException;
import javax.ejb.RemoveException;

/**
 * A Java control demonstrating how you can use multiple Java controls
 * to coordinate potentially complex business logic. This control
 * coordinates a purchase order transaction that requires a database
 * update and an EJB query. In addition to the transaction semantics of
 * this control's logic, keep in mind that the work of this control is
 * itself part of a transaction that will be rolled back if something
 * goes wrong while the control is executing.
 *
 * @jcs:jcs-jar
 * label="VerifyFunds"
 * palette-priority="3"
 * group-name="SampleControls"
 * version="1.0"
 * description="A control that contains other controls."
 * @editor-info:code-gen control-interface="true"
 */
public class VerifyFundsImpl implements VerifyFunds, com.bea.control.ControlSource
{
    public Callback callback;

    /**
     * A utility control for formatting the PO number.
     *
     * @common:control
     */
    private localControls.nestedControls.poUtil.POUtil poUtilJC;

    /**
     * An EJB control for accessing the customer's account.
     *
     * @common:control
     */
    private localControls.nestedControls.CustomerAccountEJB customerAccountEJB;

    /**
     * A Database control for accessing the item inventory.
```

navWebServices.html Sample

```

*
* @common:control
*/
private localControls.nestedControls.ItemsDatabase itemsDB;

/**
 * A global variable to hold the balance available.
 */
public boolean m_isBalanceAvailable = false;

/**
 * A global variable to hold the inventory available.
 */
    public boolean m_isInventoryAvailable = false;

/**
 * Submits a new purchase order, using an EJB to check the customer's
 * balance and a database to check item inventory. <br/>
 *
 * All of the parameter values are set by the client of this
 * control. For most, the values may be any int (although the first
 * may contain letters also). For the
 * the itemNumber and quantityRequested parameters, the value
 * matters because are needed for database queries. The item numbers
 * and quantities known to the database are stored in following table. <br/>
 *
 * <table style="font-size: 8 pt">
 * <tr><td style="width: 100px">itemNumber</td><td style="width: 100px">quantityRequested</td></tr>
 * <tr><td>624</td><td>5</td></tr>
 * <tr><td>625</td><td>6</td></tr>
 * <tr><td>629</td><td>10</td></tr>
 * <tr><td>631</td><td>15</td></tr>
 * <tr><td>640</td><td>1</td></tr>
 * </table>
 *
 * @param poNumberString The number to use for this new purchase order.
 * Any int will do here.
 * @param customerIDString The customer's unique ID. May be anything.
 * @param itemNumber A number corresponding to an item stored in the
 * database.
 * @param quantityRequested The number of items to purchase. Used to
 * query for available inventory.
 * @param startingBalance The amount of money the customer has. Used to
 * create an EJB representing the customer's account.
 *
 * @common:operation
 */
public void submitPO(String poNumberString, String customerIDString,
    int itemNumber, int quantityRequested, double startingBalance)
{
    String accountResult = this.openAccount(startingBalance, customerIDString);

    int poNumber = poUtilJC.formatNumber(poNumberString);
    int customerID = poUtilJC.formatNumber(customerIDString);

    double itemPrice = itemsDB.selectItemPrice(itemNumber);
    int quantityAvailable = itemsDB.checkInventory(itemNumber);
    double totalAmount = itemPrice * quantityRequested;

```

navWebServices.html Sample

```
try
{
    /** Get the customer's available balance. */
    double balance = customerAccountEJB.balance();

    /**
     * If the balance is greater than the request's total, set the
     * "balance available" flag to true.
     */
    if (customerAccountEJB.balance() > totalAmount)
    {
        m_isBalanceAvailable = true;
    }
    /**
     * If the inventory is greater than the request's total, set the
     * "inventory available" flag to true.
     */
    if (quantityAvailable >= quantityRequested)
    {
        m_isInventoryAvailable = true;
    }
    /**
     * If both "inventory available" and "balance available" flags are
     * true, submit the purchase order and decrement both inventory and
     * balance. If either is false, roll back the transaction.
     */
    if (m_isBalanceAvailable && m_isInventoryAvailable)
    {
        customerAccountEJB.withdraw(totalAmount);
        itemsDB.updateInventory(itemNumber, quantityAvailable - quantityRequested);
        callback.onTransactionComplete("Transaction successful.", m_isBalanceAvailable,
            m_isInventoryAvailable);
    }
    else
    {
        callback.onTransactionComplete("Unable to complete the transaction.",
            m_isBalanceAvailable, m_isInventoryAvailable);
    }

    /**
     * Call the EJB's remove method to delete the bean's record from
     * the database. This allows you to test repeatedly with the same
     * customer number.
     */
    customerAccountEJB.remove();
}
catch (RemoveException rve)
{
    throw new ControlException ("VerifyFunds: Error removing Account EJB. " +
        "Transaction canceled.", rve);
}
catch (RemoteException re)
{
    throw new ControlException ("VerifyFunds: Error getting information about the " +
        "account. Transaction canceled.", re);
}
catch (ProcessingErrorException pe)
{
    throw new ControlException ("VerifyFunds: Error withdrawing funds. " +
        "Transaction canceled.", pe);
}
```

```

    }

    /**
     * Used to create a new instance of an Account EJB representing the
     * customer's account. Purchase order details are calculated and the
     * resulting total is checked against the amount of money the customer
     * has.
     *
     * @param initialBalance The amount of money the customer can spend.
     * @param accountID The customer's unique ID.
     * @return A message indicating whether or not the account
     * was created.
     */
    private String openAccount(double initialBalance, String accountID)
    {
        String result = null;
        String accountType = "checking";

        try
        {
            Account buyerAccount = customerAccountEJB.create(accountID,
            initialBalance, accountType);
            result = "Created account: " + accountID;
        }
        catch (Exception e)
        {
            throw new RuntimeException("Create account failed for account ID:" + accountID);
        }
        return result;
    }
}

```

tags Samples

This section contains source code for the following samples.

Samples Included in This Section

ItemsDatabaseControl.jcx Sample

POVerify-tags.xml Sample

POVerify.java Sample

POVerifyImpl.jcs Sample

ItemsDatabaseControl.jcx Sample

This topic includes the source code for the ItemsDatabaseControl.jcx Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/localControls/tags/

Sample Source Code

```
package localControls.tags;

import com.bea.control.DatabaseControl;
import java.sql.SQLException;

/**
 * This Database control provides access to a database for the
 * POVerify Java control.
 *
 * @jc:connection data-source-jndi-name="cgSampleDataSource"
 */
public interface ItemsDatabaseControl
    extends DatabaseControl, com.bea.control.ControlExtension
{
    /**
     * @jc:sql statement::
     * INSERT INTO po_customers (orderid, customerid)
     * VALUES ({poNumber}, {customerID})::
     */
    void insertItemCustomer(int poNumber, int customerID);

    /**
     * @jc:sql statement::
     * INSERT INTO po_items (orderid, itemnumber, quantity)
     * VALUES ({poNumber}, {itemNumber}, {quantity})::
     */
    void insertPOItem(int poNumber, int itemNumber, int quantity);

    /**
     * @jc:sql statement="SELECT quantityAvailable FROM items WHERE itemNumber = {itemNumber}"
     */
    int checkInventory(int itemNumber);
}
```

POVerify-tags.xml Sample

This topic includes the source code for the POVerify-tags.xml Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/localControls/tags/

Sample Source Code

```
<?xml version="1.0"?>
<control-tags xmlns="http://www.bea.com/2003/03/controls/">
  <control-tag name="databaseActions">
    <description>Verifies purchase order information.</description>
    <attribute name="checkInventory" required="false">
      <description>Specifies whether to check inventory before updating the d
      <type>
        <boolean/>
      </type>
      <default-value>>false</default-value>
    </attribute>
  </control-tag>
</control-tags>
```

POVerify.java Sample

This topic includes the source code for the POverify.java Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/localControls/tags/

Sample Source Code

```
package localControls.tags;

import com.bea.control.Control;

/**
 * This is the public interface for the POverify control.
 * The control's logic is implemented in POverifyImpl.
 */
public interface POverify extends Control
{
    /**
     * As with web services built with WebLogic Workshop, a control method exposed to clients i
     * @common:operation
     */
    boolean addPurchaseOrder(java.lang.String poNumberString, int customerID, int itemNumber, i
}
```


POVerifyImpl.jcs Sample

This topic includes the source code for the POVerifyImpl.jcs Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/localControls/tags/

Sample Source Code

```
package localControls.tags;

import com.bea.control.ControlContext;

/**
 * The POVerify control is designed to
 * accept the details of an item associated with a purchase order,
 * and to insert the details into two database tables. The control
 * has the capability to make the database submission conditional based on
 * whether there is enough inventory for the item. It exposes a
 * "checkInventory" attribute through which a developer
 * using the control can request this feature.<br/><br/>
 *
 * @jcs:jc-jar
 * label="POVerify"
 * palette-priority="3"
 * group-name="SampleControls"
 * version="1.0"
 * description="Checks inventory before submitting a purchase order."
 * @editor-info:code-gen control-interface="true"
 * @jcs:control-tags file="POVerify-tags.xml"
 */
public class POVerifyImpl implements POVerify, com.bea.control.ControlSource
{
    /**
     * @common:control
     */
    private localControls.tags.ItemsDatabaseControl itemsDB;

    /**
     * @common:context
     */
    ControlContext context;

    /**
     * As with web services built with WebLogic Workshop, a control method
     * exposed to clients is annotated with a common:operation tag. The
     * addPurchaseOrder method does most of the work in this control. It is
     * the only method exposed to clients.
     *
     * @common:operation
     */
    public boolean addPurchaseOrder(String poNumberString, int customerID,
        int itemNumber, int quantity)
    {

```

navWebServices.html Sample

```
/*
 * Use the helper function at the bottom of this code to format the
 * incoming purchase order number to an int that will be accepted by the
 * database.
 */
int poNumber = formatPO(poNumberString);

/*
 * Use the ControlContext object's getControlAttribute method to retrieve the
 * checkInventory attribute's value set by the developer using the control.
 * The return type of getControlAttribute is always a String, so convert it to
 * a boolean through which you can check the value.
 */
String checkInventoryString = context.getControlAttribute("jc:databaseActions",
    "checkInventory");
boolean checkInventory = new Boolean(checkInventoryString).booleanValue();

/*
 * If the checkInventory attribute has been set to true, then
 * check inventory using the Database control before inserting the
 * rows pertaining to this request.
 */
if (checkInventory)
{
    /*
     * Use the Database control to get the number of items available.
     */
    int quantityAvailable = itemsDB.checkInventory(itemNumber);

    /*
     * If the number of items available is equal to or greater than
     * the number requested, then go ahead and update the database.
     */
    if (quantityAvailable >= quantity)
    {
        itemsDB.insertItemCustomer(poNumber, customerID);
        itemsDB.insertPOItem(poNumber, itemNumber, quantity);
        return true;
    }

    /*
     * If the number of items available is not equal to or greater
     * than the number requested, return false to indicated that the
     * update was not successful.
     */
    else
    {
        return false;
    }
}

/*
 * If the checkInventory attribute is set to false, then
 * simply perform the database inserts and return true to indicate it
 * has been done.
 */
else
{
    itemsDB.insertItemCustomer(poNumber, customerID);
    itemsDB.insertPOItem(poNumber, itemNumber, quantity);
}
```

navWebServices.html Sample

```
        return true;
    }
}

/*
 * A helper function to format the purchase order number from a String
 * type whose value contains dashes to an int.
 */
private int formatPO(String poNumber)
{
    StringBuffer bufferPONumber = new StringBuffer(poNumber);
    while (bufferPONumber.indexOf("-") >= 0)
    {
        int dashIndex = bufferPONumber.indexOf("-");
        bufferPONumber.deleteCharAt(dashIndex);
    }
    return new Integer(bufferPONumber.toString()).intValue();
}
}
```

proxy Samples

This section contains source code for the following samples.

Samples Included in This Section

mazegen Samples

register Samples

mazegen Samples

This section contains source code for the following samples.

Samples Included in This Section

MazeGenerator.jws Sample

MazeMap.jsx Sample

Partition.java Sample

MazeGenerator.jws Sample

This topic includes the source code for the MazeGenerator.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/proxy/mazegen/

Sample Source Code

```
package proxy.mazegen;

import java.util.Vector;

/**
 * This is a web service for generating random mazes. It has two entry points:
 * one returns an XML version of the maze and the other returns a pretty-printed
 * string version.
 * @common:target-namespace namespace="http://workshop.bea.com/AccountEJBClient"
 */

public class MazeGenerator implements java.io.Serializable, com.bea.jws.WebService
{

    /**
     * Returns a random maze in an XML format.
     *
     * @common:operation
     * @jws:conversation phase=none
     */
    public int [] getRandomMaze(int rows, int columns)
    {
        int r, c;
        int [] rooms = new int[rows * columns];

        Maze maze = getRandomMazeImpl(rows, columns);
        for(r = 0; r < rows; r++)
        {
            for(c = 0; c < columns; c++)
            {
                rooms[maze.toIndex(r, c)] = maze.getRoomWalls(r, c);
            }
        }
        return rooms;
    }

    /**
     * This operation generates a random maze and returns a pretty-printed string
     * representation of it.
     *
     * @common:operation
     * @jws:conversation phase="none"
     */
    public String printRandomMaze(int rows, int columns)
    {

```

navWebServices.html Sample

```
// Produce a random maze in our data structures.
Maze maze = getRandomMazeImpl(rows, columns);

// This will store the pretty-printed string. We start out with an extra
// newline so that it looks nice in the web UI.
StringBuffer buf = new StringBuffer("\n");

// Write the top row of the maze drawing. This is just a long line.

buf.append('+');

for (int i = 0; i < maze.cols; i++)
    buf.append("---");

buf.append('\n');

// Next, we draw 2 lines for each row of the maze.
for (int i = 0; i < maze.rows; i++)
{
    // Write the second middle text row of this row of the maze. This will have a separator
    // at the right end iff there is not a door between this room and the one to its right.

    buf.append('|');

    for (int j = 0; j < maze.cols; j++)
    {
        buf.append(" ");

        if ((j + 1 < maze.cols) && maze.isConnected(i, j, i, j + 1))
            buf.append(' ');
        else
            buf.append('|');
    }

    buf.append('\n');

    // Write the bottom text row of this maze row. It has a separator iff there is a door
    // between this room and the one below it.

    buf.append("+");

    for (int j = 0; j < maze.cols; j++)
    {
        if ((i + 1 < maze.rows) && maze.isConnected(i, j, i + 1, j))
            buf.append(" ");
        else
            buf.append("--");

        buf.append("+");
    }

    buf.append("\n");
}

// Return the pretty-printed string.
return buf.toString();
}

/** This is our internal datastructure for representing a maze. */
public static class Maze
```

```

{

    public static int LEFT_WALL = 1;
    public static int TOP_WALL = 2;
    public static int RIGHT_WALL = 4;
    public static int BOTTOM_WALL = 8;

    public int rows;
    public int cols;
    public boolean[][] connected;

    public Maze()
    {
        // This is necessary for using this type in an outputPart.
    }

    public Maze(int rows, int cols)
    {
        this.rows = rows;
        this.cols = cols;
        this.connected = new boolean[getIndexCount()][getIndexCount()];
    }

    public int getRows()
    {
        return rows;
    }

    public int getColumns()
    {
        return cols;
    }

    public int getIndexCount()
    {
        return rows * cols;
    }

    public int toIndex(int row, int col)
    {
        return row * cols + col;
    }

    public boolean getConnected(int row1, int col1, int row2, int col2)
    {
        return connected[toIndex(row1, col1)][toIndex(row2, col2)];
    }

    public void setConnected(int row1, int col1, int row2, int col2)
    {
        setConnected(toIndex(row1, col1), toIndex(row2, col2));
    }

    public void setConnected(int index1, int index2)
    {
        connected[index1][index2] = true;
        connected[index2][index1] = true;
    }

    public int getRoomWalls(int row, int col)
    {

```


navWebServices.html Sample

```

int walls = 0;

// see if the left wall should exist: first column or column not connected on left
if ((col == 0) ||
    ((col > 0) && !connected[toIndex(row,col)][toIndex(row,col-1)]))
{
    walls += LEFT_WALL;
}

// see if the right wall should exist: last column or column not connected on right
if ((col == (cols-1)) ||
    ((col < cols-1) && !connected[toIndex(row,col)][toIndex(row,col+1)]))
{
    walls += RIGHT_WALL;
}

// see if the bottom wall should exist: first row or row not connected on bottom
if ((row == 0) ||
    ((row > 0) && !connected[toIndex(row,col)][toIndex(row-1,col)]))
{
    walls += BOTTOM_WALL;
}

// see if the top wall should exist: last row or row not connected on top
if ((row == (rows-1)) ||
    ((row < rows-1) && !connected[toIndex(row,col)][toIndex(row+1,col)]))
{
    walls += TOP_WALL;
}

return walls;
}
}

/** Returns the actual random maze. */
private Maze getRandomMazeImpl(int rows, int cols)
{
    // Create an empty maze with no connecteness.
    Maze maze = new Maze(rows, cols);

    // Create a vector that contains all possible edges.

    Vector edges = new Vector();

    for (int i = 0; i < maze.rows; i++)
    {
        for (int j = 0; j < maze.cols; j++)
        {
            if (j + 1 < maze.cols)
                edges.add(new Edge(maze.toIndex(i, j), maze.toIndex(i, j + 1)));

            if (i + 1 < maze.rows)
                edges.add(new Edge(maze.toIndex(i, j), maze.toIndex(i + 1, j)));
        }
    }

    // Randomly permute the edges.
    for (int i = 0; i < edges.size() - 1; i++)
    {
        // Pick which edge should go at this position in the array.
        int choice = (int)((edges.size() - i) * Math.random());
    }
}

```

navWebServices.html Sample

```
// Swap the edge at this position with the one chosen.
Edge e = (Edge) edges.get(i);
edges.set(i, edges.get(choice));
edges.set(choice, e);
}

// Use a partition to determine which nodes are in the same tree.
Partition partition = new Partition(maze.getIndexCount());

// Try to add each edge in sequence, ensuring that no cycle is every produced.
for (int i = 0; i < edges.size(); i++)
{
    // Retrieve the next edge from the list.
    Edge edge = (Edge) edges.get(i);

    // If the ends of this edge are not in the same tree, then connecting them
    // will not cause a cycle.
    if (!partition.inSameGroup(edge.index1, edge.index2))
    {
        // Add this edge into the maze.
        maze.setConnected(edge.index1, edge.index2);

        // Note that these two trees are now in the same tree.
        partition.joinGroups(edge.index1, edge.index2);
    }
}

// Return the maze produced.
return maze;
}

/** Stores an undirected edge between two nodes. */
private static class Edge
{
    public int index1;
    public int index2;

    public Edge(int index1, int index2)
    {
        this.index1 = index1;
        this.index2 = index2;
    }
}
}
```

MazeMap.jsx Sample

This topic includes the source code for the MazeMap.jsx Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/proxy/mazegen/

Sample Source Code

```
package proxy.mazegen;

import proxy.mazegen.MazeGenerator.Maze;

function ConvertMazeToXML(maze)
{
    resultXML = <maze></maze>;
    resultXML.appendChild(<rows>{maze.rows}</rows>);
    resultXML.appendChild(<cols>{maze.columns}</cols>);
    rooms = <rooms></rooms>;

    for (i = 0; i < maze.rows; i++)
    {
        for (j = 0; j < maze.columns; j++)
        {
            N = hasDoor(maze, i, j, i - 1, j);
            S = hasDoor(maze, i, j, i + 1, j);
            E = hasDoor(maze, i, j, i, j - 1);
            W = hasDoor(maze, i, j, i, j + 1);

            rooms.appendChild(
                <room row={i} col={j}>
                    <hasNorthDoor>{N}</hasNorthDoor>
                    <hasSouthDoor>{S}</hasSouthDoor>
                    <hasEastDoor>{E}</hasEastDoor>
                    <hasWestDoor>{W}</hasWestDoor>
                </room>);
        }
    }

    resultXML.appendChild(rooms);
    return resultXML;
}

function hasDoor(maze, row1, col1, row2, col2)
{
    if ((row1 < 0) || (maze.rows <= row1))
        return false;

    if ((col1 < 0) || (maze.columns <= col1))
        return false;

    if ((row2 < 0) || (maze.rows <= row2))
        return false;
```

navWebServices.html Sample

```
if ((col2 < 0) || (maze.columns <= col2))  
    return false;  
  
return maze.isConnected(row1, col1, row2, col2);  
}
```

Partition.java Sample

This topic includes the source code for the Partition.java Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/proxy/mazegen/

Sample Source Code

```
package proxy.mazegen;

/**
 * This class implements a data structure for partitioning a set of numbers into groups.
 * It supports quickly determining if two numbers are in the same group and also joining
 * two groups together. Initially, each number is in its own group.
 */
public class Partition
{
    /** Stores the parent of a given node in its tree. */
    private int[] parent;

    /**
     * Creates a partition of the numbers 0 to size-1. Initially, each number is in its
     * own partition.
     */
    public Partition(int size)
    {
        parent = new int[size];

        for (int i = 0; i < parent.length; i++)
            parent[i] = i;
    }

    /** Determines if the given numbers are in the same group. */
    public boolean inSameGroup(int num1, int num2)
    {
        // These two numbers are in the same tree if they have the same root.
        return root(num1) == root(num2);
    }

    /** Returns the root of the tree containing this number. */
    private int root(int num)
    {
        // Find the root of the tree containing this number.

        int root = num;

        while (parent[root] != root)
            root = parent[root];

        // Set the parent of every node from num up in its tree to have root
        // as its parent.

        while (num != root)
    }
}
```

navWebServices.html Sample

```
{
    // Retrieve the original parent of this number.
    int next = parent[num];

    // Change the parent to point to the top of this tree.
    parent[num] = root;

    // Next, do the same to the original parent of this node.
    num = next;
}

// Return the root that was found.
return root;
}

/** Join the groups containing the two given numbers into the same group. */
public void joinGroups(int num1, int num2)
{
    // Set the parent of num2's tree to be the root of num1's tree. This
    // groups them into the same tree.
    parent[root(num2)] = root(num1);
}
}
```

register Samples

This section contains source code for the following samples.

Samples Included in This Section

Contact.java Sample

Person.java Sample

RegisterPerson.jws Sample

Contact.java Sample

This topic includes the source code for the Contact.java Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/proxy/register/

Sample Source Code

```
package proxy.register;

public class Contact implements java.io.Serializable
{
    public String street;
    public String city;
    public String state;
    public String zip;
    public String phone;

    public Contact() {}
}
```


Person.java Sample

This topic includes the source code for the Person.java Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/proxy/register/

Sample Source Code

```
package proxy.register;

public class Person implements java.io.Serializable
{
    public String name;
    public Contact home;
    public Contact work;

    public Person() {}
}
```

RegisterPerson.jws Sample

This topic includes the source code for the RegisterPerson.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/proxy/register/

Sample Source Code

```
package proxy.register;

import proxy.register.Person;
import proxy.register.Contact;

/**
 * <p>This web service serves as a target web service for the RegisterClient.java
 * web service client. The client illustrates invocation of a WebLogic Workshop
 * conversational web service that uses complex Java types in its interface.</p>
 *
 * <p>A sample Java client for this web service is located in:
 * %WL_HOME%\samples\workshop\SamplesApp\ProxyClient\register\RegisterClient.java</p>
 *
 * @jws:conversation-lifetime max-age="5 minutes"
 */
public class RegisterPerson implements com.bea.jws.WebService
{
    /**
     * m_Person represents the conversational state of this web service. Any member
     * variables of the web service's class are persisted on a per conversation
     * basis.
     */
    Person m_Person = null;

    /**
     * <p>Starts a conversation and accepts initial Person data.</p><br>
     *
     * @common:operation
     * @jws:conversation phase="start"
     */
    public boolean setPerson(Person newPerson)
    {
        boolean retval = false;

        if( newPerson != null ) {
            m_Person = newPerson;
            retval = true;
        }
        else {
            m_Person = null;
            retval = false;
        }
        return retval;
    }
}
```

navWebServices.html Sample

```
/**
 * <p>Returns the current Person data.</p><br>
 *
 * @common:operation
 * @jws:conversation phase="continue"
 */
public Person getPerson()
{
    return m_Person;
}

/**
 * <p>Accepts new <tt>home</tt> contact data for the Person.</p>
 *
 * <p>The default SOAP message encoding for WebLogic Workshop web services is
 * SOAP document/literal, but this may be set on a per web service or per
 * method basis. This method uses the default, so it expects its argument to
 * be document/literal encoded.</p><br>
 *
 * @common:operation
 * @jws:conversation phase="continue"
 */
public boolean setHomeContact(Contact newContact)
{
    boolean retval = false;

    if( newContact != null ) {
        m_Person.home = newContact;
        retval = true;
    }
    else {
        m_Person.home = null;
        retval = false;
    }
    return retval;
}

/**
 * <p>Accepts new <tt>home</tt> contact data for the Person.</p>
 *
 * <p>The default SOAP message encoding for WebLogic Workshop web services is
 * SOAP document/literal, but this may be set on a per web service or per
 * method basis. This method overrides the default, specifying it expects its
 * argument to be SOAP-RPC encoded.</p><br>
 *
 * @common:operation
 * @jws:conversation phase="continue"
 * @jws:protocol soap-style="rpc"
 */
public boolean setHomeContactRPC(Contact newContact)
{
    boolean retval = false;

    if( newContact != null ) {
        m_Person.home = newContact;
        retval = true;
    }
    else {
        m_Person.home = null;
        retval = false;
    }
}
```

navWebServices.html Sample

```
        return retval;
    }

/**
 * <p>Returns the current <tt>home</tt> contact data for the Person.</p>
 *
 * <p>The default SOAP message encoding for WebLogic Workshop web services is
 * SOAP document/literal, but this may be set on a per web service or per
 * method basis. This method uses the default, so its return value is
 * document/literal encoded.</p><br>
 *
 * @common:operation
 * @jws:conversation phase="continue"
 */
public Contact getHomeContact()
{
    Contact retval = null;

    if( m_Person.home != null ) {
        retval = m_Person.home;
    }
    return retval;
}

/**
 * <p>Accepts new <tt>work</tt> contact data for the Person.</p>
 *
 * <p>The default SOAP message encoding for WebLogic Workshop web services is
 * SOAP document/literal, but this may be set on a per web service or per
 * method basis. This method uses the default, so it expects its argument to
 * be document/literal encoded.</p><br>
 *
 * @common:operation
 * @jws:conversation phase="continue"
 */
public boolean setWorkContact(Contact newContact)
{
    boolean retval = false;

    if( newContact != null ) {
        m_Person.work = newContact;
        retval = true;
    }
    else {
        m_Person.work = null;
        retval = false;
    }
    return retval;
}

/**
 * <p>Accepts new <tt>work</tt> contact data for the Person.</p>
 *
 * <p>The default SOAP message encoding for WebLogic Workshop web services is
 * SOAP document/literal, but this may be set on a per web service or per
 * method basis. This method overrides the default, specifying it expects its
 * argument to be SOAP-RPC encoded.</p><br>
 *
 * @common:operation
 * @jws:conversation phase="continue"
 * @jws:protocol soap-style="rpc"

```

navWebServices.html Sample

```
*/
public boolean setWorkContactRPC(Contact newContact)
{
    boolean retval = false;

    if( newContact != null ) {
        m_Person.work = newContact;
        retval = true;
    }
    else {
        m_Person.work = null;
        retval = false;
    }
    return retval;
}

/**
 * <p>Returns the current <tt>work</tt> contact data for the Person.</p>
 *
 * <p>The default SOAP message encoding for WebLogic Workshop web services is
 * SOAP document/literal, but this may be set on a per web service or per
 * method basis. This method uses the default, so its return value is
 * document/literal encoded.</p><br>
 *
 * @common:operation
 * @jws:conversation phase="continue"
 */
public Contact getWorkContact()
{
    Contact retval = null;

    if( m_Person.work != null ) {
        retval = m_Person.work;
    }
    return retval;
}

/**
 * <p>Terminates the current conversation.</p>
 *
 * @common:operation
 * @jws:conversation phase="finish"
 */
public void endSession()
{
}
}
```

security Samples

This section contains source code for the following samples.

Samples Included in This Section

roleBased Samples

transport Samples

wsse Samples

roleBased Samples

This section contains source code for the following samples.

Samples Included in This Section

Bank.jws Sample

BankControl.jcx Sample

createUser Samples

VeriCheck.jws Sample

Bank.jws Sample

This topic includes the source code for the Bank.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/security/roleBased/

Sample Source Code

```
/**
 * <p> The Bank web service is used in conjunction with the VeriCheck web service.
 * VeriCheck calls the Bank service to verify that a given checking account has
 * enough money in it to cover a given check.</p>
 */

package security.roleBased;

/**
 * @common:target-namespace namespace="http://workshop.bea.com/Bank"
 */
public class Bank implements com.bea.jws.WebService
{
    public Callback callback;

    public interface Callback extends com.bea.control.ServiceControl
    {
        /**
         * @jws:conversation phase="finish"
         */
        public void onResultComplete(String accountID, String resultMessage);
    }

    /**
     * @common:operation
     * @jws:conversation phase="start"
     * @jws:message-buffer enable="true"
     */
    public void doesAccountHaveSufficientBalance(String accountID, int amount)
    {
        callback.setUsername("weblogic");
        callback.setPassword("weblogic");
        if(amount > 1000)
        {
            callback.onResultComplete(accountID, "The account does not have sufficient balance.");
        }
        else
        {
            callback.onResultComplete(accountID, "The account has sufficient balance.");
        }
    }
}
```


BankControl.jcx Sample

This topic includes the source code for the BankControl.jcx Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/security/roleBased/

Sample Source Code

```
package security.roleBased;

/**
 * @jc:location http-url="https://localhost:7002/WebServices/security/roleBased/Bank.jws"
 * @jc:wSDL file="#BankWSDL"
 */
public interface BankControl extends com.bea.control.ControlExtension, com.bea.control.ServiceControl
{
    public static class StartHeader
        implements java.io.Serializable
    {
        public java.lang.String conversationID;
        public java.lang.String callbackLocation;
    }

    public static class ContinueHeader
        implements java.io.Serializable
    {
        public java.lang.String conversationID;
    }

    public static class CallbackHeader
        implements java.io.Serializable
    {
        public java.lang.String conversationID;
    }

    public interface Callback extends com.bea.control.ServiceControl.Callback
    {
        /**
         * @jc:conversation phase="finish"
         */
        public void onResultComplete (java.lang.String accountID, java.lang.String resultMessage);
    }

    /**
     * @jc:conversation phase="start"
     */
    public void doesAccountHaveSufficientBalance (java.lang.String accountID, int amount);

    static final long serialVersionUID = 1L;
}
```

navWebServices.html Sample

```

/** @common:define name="BankWsd1" value::
<?xml version="1.0" encoding="utf-8"?>
<!-- @editor-info:link autogen="true" source="Bank.jws" -->
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:conv="http://www.openuri.org/2002/02/02/conv"
<types>
  <s:schema elementFormDefault="qualified" targetNamespace="http://workshop.bea.com/Bank">
    <s:element name="onResultCompleteResponse">
      <s:complexType>
        <s:sequence/>
      </s:complexType>
    </s:element>
    <s:element name="onResultComplete">
      <s:complexType>
        <s:sequence>
          <s:element name="accountID" type="s:string" minOccurs="0"/>
          <s:element name="resultMessage" type="s:string" minOccurs="0"/>
        </s:sequence>
      </s:complexType>
    </s:element>
    <s:element name="doesAccountHaveSufficientBalance">
      <s:complexType>
        <s:sequence>
          <s:element name="accountID" type="s:string" minOccurs="0"/>
          <s:element name="amount" type="s:int"/>
        </s:sequence>
      </s:complexType>
    </s:element>
    <s:element name="doesAccountHaveSufficientBalanceResponse">
      <s:complexType>
        <s:sequence/>
      </s:complexType>
    </s:element>
  </s:schema>

  <s:schema elementFormDefault="qualified" targetNamespace="http://www.openuri.org/2002/02/02/conv">
    <s:element name="StartHeader" type="conv:StartHeader"/>
    <s:element name="ContinueHeader" type="conv:ContinueHeader"/>
    <s:element name="CallbackHeader" type="conv:CallbackHeader"/>
    <s:complexType name="StartHeader">
      <s:sequence>
        <s:element minOccurs="0" maxOccurs="1" name="conversationID" type="s:string"/>
        <s:element minOccurs="0" maxOccurs="1" name="callbackLocation" type="s:string"/>
      </s:sequence>
    </s:complexType>
    <s:complexType name="ContinueHeader">
      <s:sequence>
        <s:element minOccurs="1" maxOccurs="1" name="conversationID" type="s:string"/>
      </s:sequence>
    </s:complexType>
    <s:complexType name="CallbackHeader">
      <s:sequence>
        <s:element minOccurs="1" maxOccurs="1" name="conversationID" type="s:string"/>
      </s:sequence>
    </s:complexType>
  </s:schema>
</types>
<message name="onResultCompleteSoapIn">
  <part name="parameters" element="s0:onResultCompleteResponse"/>
</message>
<message name="onResultCompleteSoapOut">
  <part name="parameters" element="s0:onResultComplete"/>

```

navWebServices.html Sample

```
</message>
<message name="doesAccountHaveSufficientBalanceSoapIn">
  <part name="parameters" element="s0:doesAccountHaveSufficientBalance"/>
</message>
<message name="doesAccountHaveSufficientBalanceSoapOut">
  <part name="parameters" element="s0:doesAccountHaveSufficientBalanceResponse"/>
</message>
<message name="onResultCompleteHttpGetIn"/>
<message name="onResultCompleteHttpGetOut">
  <part name="accountID" type="s:string"/>
  <part name="resultMessage" type="s:string"/>
</message>
<message name="doesAccountHaveSufficientBalanceHttpGetIn">
  <part name="accountID" type="s:string"/>
  <part name="amount" type="s:string"/>
</message>
<message name="doesAccountHaveSufficientBalanceHttpGetOut"/>
<message name="onResultCompleteHttpPostIn"/>
<message name="onResultCompleteHttpPostOut">
  <part name="accountID" type="s:string"/>
  <part name="resultMessage" type="s:string"/>
</message>
<message name="doesAccountHaveSufficientBalanceHttpPostIn">
  <part name="accountID" type="s:string"/>
  <part name="amount" type="s:string"/>
</message>
<message name="doesAccountHaveSufficientBalanceHttpPostOut"/>
<message name="StartHeader_literal">
  <part name="StartHeader" element="conv:StartHeader"/>
</message>
<message name="CallbackHeader_literal">
  <part name="CallbackHeader" element="conv:CallbackHeader"/>
</message>
<portType name="BankSoap">
  <operation name="onResultComplete">
    <output message="s0:onResultCompleteSoapOut"/>
    <input message="s0:onResultCompleteSoapIn"/>
  </operation>
  <operation name="doesAccountHaveSufficientBalance">
    <input message="s0:doesAccountHaveSufficientBalanceSoapIn"/>
    <output message="s0:doesAccountHaveSufficientBalanceSoapOut"/>
  </operation>
</portType>
<portType name="BankHttpGet">
  <operation name="onResultComplete">
    <output message="s0:onResultCompleteHttpGetOut"/>
    <input message="s0:onResultCompleteHttpGetIn"/>
  </operation>
  <operation name="doesAccountHaveSufficientBalance">
    <input message="s0:doesAccountHaveSufficientBalanceHttpGetIn"/>
    <output message="s0:doesAccountHaveSufficientBalanceHttpGetOut"/>
  </operation>
</portType>
<portType name="BankHttpPost">
  <operation name="onResultComplete">
    <output message="s0:onResultCompleteHttpPostOut"/>
    <input message="s0:onResultCompleteHttpPostIn"/>
  </operation>
  <operation name="doesAccountHaveSufficientBalance">
    <input message="s0:doesAccountHaveSufficientBalanceHttpPostIn"/>
    <output message="s0:doesAccountHaveSufficientBalanceHttpPostOut"/>
  </operation>
</portType>
```

navWebServices.html Sample

```

    </operation>
</portType>
<binding name="BankSoap" type="s0:BankSoap">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
  <operation name="onResultComplete">
    <soap:operation soapAction="http://workshop.bea.com/Bank/onResultComplete" style="document"/>
    <cw:transition phase="finish"/>
    <input>
      <soap:body use="literal"/>
    </input>
    <output>
      <soap:body use="literal"/>
      <soap:header wsdl:required="true" message="s0:CallbackHeader_literal" part="CallbackHeader"/>
    </output>
  </operation>
  <operation name="doesAccountHaveSufficientBalance">
    <soap:operation soapAction="http://workshop.bea.com/Bank/doesAccountHaveSufficientBalance" style="document"/>
    <cw:transition phase="start"/>
    <input>
      <soap:body use="literal"/>
      <soap:header wsdl:required="true" message="s0:StartHeader_literal" part="StartHeader"/>
    </input>
    <output>
      <soap:body use="literal"/>
    </output>
  </operation>
</binding>
<binding name="BankHttpGet" type="s0:BankHttpGet">
  <http:binding verb="GET"/>
  <operation name="onResultComplete">
    <http:operation location="/onResultComplete"/>
    <cw:transition phase="finish"/>
    <input>
      <mime:mimeType part="Body"/>
    </input>
    <output>
      <http:contentType"/>
    </output>
  </operation>
  <operation name="doesAccountHaveSufficientBalance">
    <http:operation location="/doesAccountHaveSufficientBalance"/>
    <cw:transition phase="start"/>
    <input>
      <http:contentType"/>
    </input>
    <output/>
  </operation>
</binding>
<binding name="BankHttpPost" type="s0:BankHttpPost">
  <http:binding verb="POST"/>
  <operation name="onResultComplete">
    <http:operation location="/onResultComplete"/>
    <cw:transition phase="finish"/>
    <input>
      <mime:mimeType part="Body"/>
    </input>
    <output>
      <mime:contentType type="application/x-www-form-urlencoded"/>
    </output>
  </operation>
  <operation name="doesAccountHaveSufficientBalance">

```

navWebServices.html Sample

```
<http:operation location="/doesAccountHaveSufficientBalance"/>
<cw:transition phase="start"/>
<input>
  <mime:content type="application/x-www-form-urlencoded"/>
</input>
<output/>
</operation>
</binding>
<service name="Bank">
  <port name="BankSoap" binding="s0:BankSoap">
    <soap:address location="http://localhost:7001/WebServices/security/roleBased/Bank.jws">
    </port>
  <port name="BankHttpGet" binding="s0:BankHttpGet">
    <http:address location="http://localhost:7001/WebServices/security/roleBased/Bank.jws">
    </port>
  <port name="BankHttpPost" binding="s0:BankHttpPost">
    <http:address location="http://localhost:7001/WebServices/security/roleBased/Bank.jws">
    </port>
  </service>
</definitions>
* ::
*/
```

createUser Samples

This section contains source code for the following samples.

Samples Included in This Section

CreateUser.jws Sample

CreateUser.jws Sample

This topic includes the source code for the CreateUser.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/security/roleBased/createUser/

Sample Source Code

```
package security.roleBased.createUser;

import java.util.Vector;
import weblogic.management.Helper;
import weblogic.management.MBeanHome;
import weblogic.management.WebLogicObjectName;
import weblogic.management.configuration.SecurityConfigurationMBean;
import weblogic.management.security.authentication.AuthenticationProviderMBean;
import weblogic.management.security.authentication.UserEditorMBean;
import weblogic.management.security.authentication.UserReaderMBean;
import weblogic.security.providers.authentication.DefaultAuthenticatorMBean;

/**
 * The following web service demonstrates the attributes
 *
 * common:security run-as-principal
 *
 * and
 *
 * common:security run-as
 *
 * Note that the user weblogic and the role Administrators are pre-defined in the default
 * authentication provider in the WebLogic Server security framework.
 *
 * The attributes @common:security run-as-principal="weblogic" run-as="Administrators" cause the
 * following elements to be written to the deployment descriptor weblogic-ejb.jar:
 *
 * <security-role-assignment>
 *   <role-name>Administrators</role-name>
 *   <externally-defined/>
 * </security-role-assignment>
 *
 * @common:security run-as-principal="weblogic" run-as="Administrators"
 * @common:target-namespace namespace="http://openuri.org/bea/samples/workshop/WebServices/security"
 */
public class createUser implements com.bea.jws.WebService
{
    static final long serialVersionUID = 1L;

    /**
     * <p>This method lists all of the users in each authentication provider used by WebLogic S
     *
     * @common:operation
     */
}
```

navWebServices.html Sample

```
public Vector listUsers()
{
    Vector users = new Vector();

    MBeanHome adminHome;

    String url = "http://127.0.0.1:7001";

    adminHome = (MBeanHome)Helper.getMBeanHome("weblogic","weblogic",url,"cgServer");

    AuthenticationProviderMBean[] providers = adminHome.getActiveDomain().getSecurityConfiguratio

    for (int i=0; providers != null && i <providers.length; i++)
    {
        if (providers[i] instanceof UserReaderMBean)
        {
            UserReaderMBean reader = (UserReaderMBean)providers[i];

            try
            {
                String cursor = reader.listUsers("*",100);

                while (reader.haveCurrent(cursor))
                {
                    users.add(reader.getCurrentName(cursor));
                    reader.advance(cursor);
                }

            }
            catch (Exception e)
            {
                e.printStackTrace();
            }
        }
    }

    return users;
}

/**
 * This method adds a user to WebLogic Server's default authenticator
 *
 * <p>The password must be 8 or more characters long.
 *
 * @common:operation
 */
public void addUser(String username, String password)
{
    MBeanHome adminHome;

    String url = "http://127.0.0.1:7001" ;

    adminHome = (MBeanHome)Helper.getMBeanHome("weblogic","weblogic",url,"cgServer");

    //providers[0] is the default authenticator
    AuthenticationProviderMBean[] providers = adminHome.getActiveDomain().getSecurityConfiguratio

    UserEditorMBean editor = (UserEditorMBean)providers[0];

    try
    {
```


navWebServices.html Sample

```
        editor.createUser(username, password, "This user created by the createUser web serv
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
}

/**
 * This method deletes a user from WebLogic Server's default authenticator.
 *
 * @common:operation
 */
public void deleteUser(String username)
{
    MBeanHome adminHome;

    String url = "http://127.0.0.1:7001" ;

    adminHome = (MBeanHome)Helper.getMBeanHome("weblogic","weblogic",url,"cgServer");

    AuthenticationProviderMBean[] providers = adminHome.getActiveDomain().getSecurityConfig

    //providers[0] is the default authenticator
    UserEditorMBean editor = (UserEditorMBean)providers[0];

    try
    {
        editor.removeUser(username);
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
}
}
```

VeriCheck.jws Sample

This topic includes the source code for the VeriCheck.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/security/roleBased/

Sample Source Code

```
/**
 * <p>The VeriCheck web service demonstrates the use of one-way SSL and basic authentication.
 * Processing begins when a client invokes the VeriCheck
 * web service through the checkForSufficientBalance method. VeriCheck then calls
 * the Bank web service to see if the specified checking account has sufficient
 * balance to cover the specified amount. When the Bank web service has completed
 * its results, it calls back the VeriCheck service via the callback onResultComplete.
 * Finally, the VeriCheck service calls back the original client through the callback
 * onCheckDone.</p>
 *
 * The Bank web service is restricted to users granted the BankUsers role.
 *
 * <p><b>web.xml</b> associates particular users with groups. The user weblogic
 * is declared as a member of the BankUsers group. This allows weblogic to pass the
 * servlet security gate when it invokes the Bank's method doesAccountHaveSufficientBalance.
 */

package security.roleBased;

import async.HelloWorldAsyncControl;

/**
 * @common:target-namespace namespace="http://workshop.bea.com/VeriCheck"
 */
public class VeriCheck implements com.bea.jws.WebService
{
    /**
     * @common:control
     */
    private security.roleBased.BankControl bankControl;

    public Callback callback;

    public interface Callback
    {
        /**
         * @jws:conversation phase="finish"
         */
        public String onCheckDone(String checkingAccountID, String resultMessage);
    }

    /**
     * <p>The VeriCheck web service takes requests for check verification from merchants.
     *
     * <p>The checking account number and the amount on the check are passed to the Bank

```

navWebServices.html Sample

```
* web service. The Bank web service returns a message saying whether or not there is
* enough money in the account to cover the check.
*
* <p>Enter any String for the <b>checkingAccountID</b> parameter.
*
* <p>Enter any integer for the <b>amount</b> parameter.
*
* @common:operation
* @jws:conversation phase="start"
* @jws:message-buffer enable="true"
*/
public void checkForSufficientBalance(String checkingAccountID, int amount)
{
    /*
    * Use the username and passwords assigned to the VeriCheck web service by the Bank.
    * These allow VeriCheck to pass the Bank's security gate.
    * 'weblogic' has been granted the role BankUser.
    */
    bankControl.setUsername("weblogic");
    bankControl.setPassword("weblogic");
    bankControl.doesAccountHaveSufficientBalance(checkingAccountID, amount);
}

public void bankControl_onResultComplete(java.lang.String accountID, java.lang.String result)
{
    callback.onCheckDone(accountID, resultMessage);
}
}
```

transport Samples

This section contains source code for the following samples.

Samples Included in This Section

basicAuthentication Samples

clientCert Samples

helloWorldSecure Samples

basicAuthentication Samples

This section contains source code for the following samples.

Samples Included in This Section

BasicAuthentication.jws Sample

BasicAuthentication.jws Sample

This topic includes the source code for the BasicAuthentication.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/security/transport/basicAuthentication/

Sample Source Code

```
package security.transport.basicAuthentication;

/**
 * To run this sample start WebLogic Server in the Workshop domain, and point
 * a browser at the following URL:
 *
 * https://localhost:7002/WebServices/security/transport/basicAuthentication/BasicAuthentication/
 *
 * When you are presented with a security certificate from the server, click "Yes".
 *
 * When you are asked for username and password, enter
 *
 * User name: weblogic
 * Password: weblogic
 *
 * @common:target-namespace namespace="http://workshop.bea.com/BasicAuthentication"
 */
public class BasicAuthentication implements com.bea.jws.WebService
{
    /**
     * @common:operation
     */
    public String hello()
    {
        return "Hello, Authenticated World!";
    }
}
```

clientCert Samples

This section contains source code for the following samples.

Samples Included in This Section

WebServiceA.jws Sample

WebServiceB.jws Sample

WebServiceBControl.jcx Sample

WebServiceA.jws Sample

This topic includes the source code for the WebServiceA.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/security/transport/clientCert/

Sample Source Code

```
package security.transport.clientCert;

/**
 * A simple web service that demonstrates how to send a client certificate.
 *
 * Note client certificates are enabled in SamplesApp, but they
 * are not enforced. For this reason, WebServiceA can still successfully
 * invoke WebServiceB (and receive a callback from WebServiceB), even if WebServiceA
 * does not succeed in sending a client certificate to WebServiceB.
 */

import java.io.File;
import weblogic.Home;

/**
 * @common:target-namespace namespace="http://openuri.org/bea/samples/workshop/clientcert/webse
 */
public class WebServiceA implements com.bea.jws.WebService
{
    /** @common:control */
    security.transport.clientCert.WebServiceBControl ctrl;

    /**
     * @common:operation
     */
    public void invokeWebServiceB()
    {
        /**
         * Enable client certificates for this web service.
         */
        ctrl.useClientKeySSL( true );

        /**
         * Specify the location and password for the keystore where the client certificates res
         *
         * The following method call to setKeystore is, strictly speaking,
         * unnecessary, since SamplesApp is already configured to use the
         * default keystore DemoIdentity.jks.
         * The call to the setKeystore() method is included to show how you would set the
         * location for another, non-default keystore.
         */
        ctrl.setKeystore(Home.getPath() + "/lib/DemoIdentity.jks", "DemoIdentityKeyStorePassPhr

        /**
         * Specify the alias in the keystore for both the client SSL certificate
```


navWebServices.html Sample

```
* and the client private key. (The certificate and the private key must
* be stored under the same alias in the same keystore.)
* The second parameter specifies the password required to access
* the keystore.
*/
ctrl.setClientCert("DemoIdentity", "DemoIdentityPassPhrase");

/**
 * Invoke the requestCallback method on WebServiceB.
 */
ctrl.requestCallback("WebServiceA");
}

public void ctrl_result(java.lang.String message)
{
    /**
     * In a real world example, WebServiceA would do something with
     * the callback recieved from WebServiceB.
     */
}
}
```

WebServiceB.jws Sample

This topic includes the source code for the WebServiceB.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/security/transport/clientCert/

Sample Source Code

```
package security.transport.clientCert;

import com.bea.control.ServiceControl;
import com.bea.control.TimerControl;
import com.bea.control.TimerControlFactory;
import java.io.File;
import weblogic.Home;

/**
 * @common:target-namespace namespace="http://openuri.org/bea/samples/workshop/clientcert/webse
 */
public class WebServiceB implements com.bea.jws.WebService
{
    public Callback callback;

    public interface Callback extends ServiceControl
    {
        /**
         * @jws:conversation phase="finish"
         * @common:message-buffer enable="true"
         */
        void result(String message);
    }

    /**
     * @common:operation
     * @common:message-buffer enable="true"
     * @jws:conversation phase="start"
     */
    public void requestCallback(String caller)
    {
        callback.result("Here is your callback, " + caller + ".");
    }
}
```

WebServiceBControl.jcx Sample

This topic includes the source code for the WebServiceBControl.jcx Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/security/transport/clientCert/

Sample Source Code

```
package security.transport.clientCert;

/**
 * @jc:location http-url="WebServiceB.jws" jms-url="WebServiceB.jws"
 * @jc:wSDL file="#WebServiceBWSDL"
 * @editor-info:link autogen-style="java" source="WebServiceB.jws" autogen="true"
 */
public interface WebServiceBControl extends com.bea.control.ControlExtension, com.bea.control.S
{
    public static class ContinueHeader
        implements java.io.Serializable
    {
        public java.lang.String conversationID;
    }

    public static class CallbackHeader
        implements java.io.Serializable
    {
        public java.lang.String conversationID;
    }

    public static class StartHeader
        implements java.io.Serializable
    {
        public java.lang.String conversationID;
        public java.lang.String callbackLocation;
    }

    public interface Callback extends com.bea.control.ServiceControl.Callback
    {
        /**
         * @jc:conversation phase="finish"
         */
        public void result (java.lang.String message);
    }

    /**
     * @jc:conversation phase="start"
     */
    public void requestCallback (java.lang.String caller);

    static final long serialVersionUID = 1L;
}
```

navWebServices.html Sample

```

/** @common:define name="WebServiceBWsd1" value::
<?xml version="1.0" encoding="utf-8"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:conv="http://www.openuri.org/2002/0
<types>
  <s:schema elementFormDefault="qualified" targetNamespace="http://openuri.org/bea/sample
    <s:element name="resultResponse">
      <s:complexType>
        <s:sequence/>
      </s:complexType>
    </s:element>
    <s:element name="result">
      <s:complexType>
        <s:sequence>
          <s:element name="message" type="s:string" minOccurs="0"/>
        </s:sequence>
      </s:complexType>
    </s:element>
    <s:element name="requestCallback">
      <s:complexType>
        <s:sequence>
          <s:element name="caller" type="s:string" minOccurs="0"/>
        </s:sequence>
      </s:complexType>
    </s:element>
    <s:element name="requestCallbackResponse">
      <s:complexType>
        <s:sequence/>
      </s:complexType>
    </s:element>
  </s:schema>

  <s:schema elementFormDefault="qualified" targetNamespace="http://www.openuri.org/2002/0
    <s:element name="StartHeader" type="conv:StartHeader"/>
    <s:element name="ContinueHeader" type="conv:ContinueHeader"/>
    <s:element name="CallbackHeader" type="conv:CallbackHeader"/>
    <s:complexType name="StartHeader">
      <s:sequence>
        <s:element minOccurs="0" maxOccurs="1" name="conversationID" type="s:string"/>
        <s:element minOccurs="0" maxOccurs="1" name="callbackLocation" type="s:string"/>
      </s:sequence>
    </s:complexType>
    <s:complexType name="ContinueHeader">
      <s:sequence>
        <s:element minOccurs="1" maxOccurs="1" name="conversationID" type="s:string"/>
      </s:sequence>
    </s:complexType>
    <s:complexType name="CallbackHeader">
      <s:sequence>
        <s:element minOccurs="1" maxOccurs="1" name="conversationID" type="s:string"/>
      </s:sequence>
    </s:complexType>
  </s:schema>
</types>
<message name="resultSoapIn">
  <part name="parameters" element="s0:resultResponse"/>
</message>
<message name="resultSoapOut">
  <part name="parameters" element="s0:result"/>
</message>
<message name="requestCallbackSoapIn">

```

navWebServices.html Sample

```
<part name="parameters" element="s0:requestCallback"/>
</message>
<message name="requestCallbackSoapOut">
  <part name="parameters" element="s0:requestCallbackResponse"/>
</message>
<message name="resultHttpGetIn"/>
<message name="resultHttpGetOut">
  <part name="message" type="s:string"/>
</message>
<message name="requestCallbackHttpGetIn">
  <part name="caller" type="s:string"/>
</message>
<message name="requestCallbackHttpGetOut"/>
<message name="resultHttpPostIn"/>
<message name="resultHttpPostOut">
  <part name="message" type="s:string"/>
</message>
<message name="requestCallbackHttpPostIn">
  <part name="caller" type="s:string"/>
</message>
<message name="requestCallbackHttpPostOut"/>
<message name="StartHeader_literal">
  <part name="StartHeader" element="conv:StartHeader"/>
</message>
<message name="CallbackHeader_literal">
  <part name="CallbackHeader" element="conv:CallbackHeader"/>
</message>
<portType name="WebServiceBSoap">
  <operation name="result">
    <output message="s0:resultSoapOut"/>
    <input message="s0:resultSoapIn"/>
  </operation>
  <operation name="requestCallback">
    <input message="s0:requestCallbackSoapIn"/>
    <output message="s0:requestCallbackSoapOut"/>
  </operation>
</portType>
<portType name="WebServiceBHttpGet">
  <operation name="result">
    <output message="s0:resultHttpGetOut"/>
    <input message="s0:resultHttpGetIn"/>
  </operation>
  <operation name="requestCallback">
    <input message="s0:requestCallbackHttpGetIn"/>
    <output message="s0:requestCallbackHttpGetOut"/>
  </operation>
</portType>
<portType name="WebServiceBHttpPost">
  <operation name="result">
    <output message="s0:resultHttpPostOut"/>
    <input message="s0:resultHttpPostIn"/>
  </operation>
  <operation name="requestCallback">
    <input message="s0:requestCallbackHttpPostIn"/>
    <output message="s0:requestCallbackHttpPostOut"/>
  </operation>
</portType>
<binding name="WebServiceBSoap" type="s0:WebServiceBSoap">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
  <operation name="result">
    <soap:operation soapAction="http://openuri.org/bea/samples/workshop/clientcert/webser
```

navWebServices.html Sample

```
<cw:transition phase="finish"/>
<input>
  <soap:body use="literal"/>
</input>
<output>
  <soap:body use="literal"/>
  <soap:header wsdl:required="true" message="s0:CallbackHeader_literal" part="CallbackHeader"/>
</output>
</operation>
<operation name="requestCallback">
  <soap:operation soapAction="http://openuri.org/bea/samples/workshop/clientcert/webse
  <cw:transition phase="start"/>
  <input>
    <soap:body use="literal"/>
    <soap:header wsdl:required="true" message="s0:StartHeader_literal" part="StartHeader"/>
  </input>
  <output>
    <soap:body use="literal"/>
  </output>
</operation>
</binding>
<binding name="WebServiceBHttpGet" type="s0:WebServiceBHttpGet">
  <http:binding verb="GET"/>
  <operation name="result">
    <http:operation location="/result"/>
    <cw:transition phase="finish"/>
    <input>
      <mime:mimeXml part="Body"/>
    </input>
    <output>
      <http:urlEncoded/>
    </output>
  </operation>
  <operation name="requestCallback">
    <http:operation location="/requestCallback"/>
    <cw:transition phase="start"/>
    <input>
      <http:urlEncoded/>
    </input>
    <output/>
  </operation>
</binding>
<binding name="WebServiceBHttpPost" type="s0:WebServiceBHttpPost">
  <http:binding verb="POST"/>
  <operation name="result">
    <http:operation location="/result"/>
    <cw:transition phase="finish"/>
    <input>
      <mime:mimeXml part="Body"/>
    </input>
    <output>
      <mime:content type="application/x-www-form-urlencoded"/>
    </output>
  </operation>
  <operation name="requestCallback">
    <http:operation location="/requestCallback"/>
    <cw:transition phase="start"/>
    <input>
      <mime:content type="application/x-www-form-urlencoded"/>
    </input>
    <output/>
  </operation>
</binding>
```

navWebServices.html Sample

```
</operation>
</binding>
<service name="WebServiceB">
  <port name="WebServiceBSoap" binding="s0:WebServiceBSoap">
    <soap:address location="http://localhost:7001/security/transport/clientCert/WebServiceBSoap" />
  </port>
  <port name="WebServiceBHttpGet" binding="s0:WebServiceBHttpGet">
    <http:address location="http://localhost:7001/security/transport/clientCert/WebServiceBHttpGet" />
  </port>
  <port name="WebServiceBHttpPost" binding="s0:WebServiceBHttpPost">
    <http:address location="http://localhost:7001/security/transport/clientCert/WebServiceBHttpPost" />
  </port>
</service>
</definitions>
* ::
*/
```

helloWorldSecure Samples

This section contains source code for the following samples.

Samples Included in This Section

HelloWorldSecure.jws Sample

HelloWorldSecureClient.jws Sample

HelloWorldSecureControl.jcx Sample

HelloWorldSecure.jws Sample

This topic includes the source code for the HelloWorldSecure.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/security/transport/helloWorldSecure/

Sample Source Code

```
/**
 * This web service is secured with one-way SSL, basic authentication, and role-based security.
 * To run this web service see the HelloWorldSecureClient.jws web service.
 */
package security.transport.helloWorldSecure;

/**
 * @common:target-namespace namespace="http://workshop.bea.com/HelloWorldSecure"
 */
public class HelloWorldSecure implements com.bea.jws.WebService
{
    /**
     * @common:operation
     */
    public String HelloWorldSecure()
    {
        return "Hello, Friends!";
    }
}
```

HelloWorldSecureClient.jws Sample

This topic includes the source code for the HelloWorldSecureClient.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/security/transport/helloWorldSecure/

Sample Source Code

```
/**
 * <p>This web service serves as a client to the HelloWorldSecure web service.</p>
 */

package security.transport.helloWorldSecure;

/**
 * @common:target-namespace namespace="http://workshop.bea.com/HelloWorldSecureClient"
 */
public class HelloWorldSecureClient implements com.bea.jws.WebService
{
    /**
     * @common:control
     */
    private security.transport.helloWorldSecure.HelloWorldSecureControl hws;

    /**
     * <p>Only users granted the role of Friends can access the web service
     * HelloWorldSecure.jws. (See the <security-constraint> element in
     * WEB-INF/web.xml.
     *
     * <p>The user "weblogic", a pre-configured administrative user in
     * WebLogic Server, has been granted the role of Friend. (See WEB-INF/weblogic.xml.)
     *
     * <p>This user's authentication information (username and password)
     * are passed to the HelloWorldSecure web service via the method calls setUsername() and
     * setPassword().
     *
     * @common:operation
     */
    public String invokeHWSecure()
    {
        hws.setUsername("weblogic");
        hws.setPassword("weblogic");
        String str = hws.HelloWorldSecure();
        return str;
    }
}
```

HelloWorldSecureControl.jcx Sample

This topic includes the source code for the HelloWorldSecureControl.jcx Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/security/transport/helloWorldSecure/

Sample Source Code

```
package security.transport.helloWorldSecure;

/**
 * @jc:location http-url="https://localhost:7002/WebServices/security/transport/helloWorldSecure"
 * @jc:wSDL file="#HelloWorldSecureWSDL"
 */
public interface HelloWorldSecureControl extends com.bea.control.ControlExtension, com.bea.control.Control {

    public java.lang.String HelloWorldSecure ();

    static final long serialVersionUID = 1L;
}

/** @common:define name="HelloWorldSecureWSDL" value::
    <?xml version="1.0" encoding="utf-8"?>
    <!-- @editor-info:link autogen="true" source="HelloWorldSecure.jws" -->
    <definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:conv="http://www.openuri.org/2003/11/conv"
        <types>
            <s:schema elementFormDefault="qualified" targetNamespace="http://workshop.bea.com/HelloWorldSecure">
                <s:element name="HelloWorldSecure">
                    <s:complexType>
                        <s:sequence/>
                    </s:complexType>
                </s:element>
                <s:element name="HelloWorldSecureResponse">
                    <s:complexType>
                        <s:sequence>
                            <s:element name="HelloWorldSecureResult" type="s:string" minOccurs="0"/>
                        </s:sequence>
                    </s:complexType>
                </s:element>
                <s:element name="string" nillable="true" type="s:string"/>
            </s:schema>

        </types>
        <message name="HelloWorldSecureSoapIn">
            <part name="parameters" element="s0:HelloWorldSecure"/>
        </message>
        <message name="HelloWorldSecureSoapOut">
            <part name="parameters" element="s0:HelloWorldSecureResponse"/>
        </message>
        <message name="HelloWorldSecureHttpGetIn">
        <message name="HelloWorldSecureHttpGetOut">
            <part name="Body" element="s0:string"/>
        </message>
    </definitions>
```

navWebServices.html Sample

```

</message>
<message name="HelloWorldSecureHttpPostIn"/>
<message name="HelloWorldSecureHttpPostOut">
  <part name="Body" element="s0:string"/>
</message>
<portType name="HelloWorldSecureSoap">
  <operation name="HelloWorldSecure">
    <input message="s0:HelloWorldSecureSoapIn"/>
    <output message="s0:HelloWorldSecureSoapOut"/>
  </operation>
</portType>
<portType name="HelloWorldSecureHttpGet">
  <operation name="HelloWorldSecure">
    <input message="s0:HelloWorldSecureHttpGetIn"/>
    <output message="s0:HelloWorldSecureHttpGetOut"/>
  </operation>
</portType>
<portType name="HelloWorldSecureHttpPost">
  <operation name="HelloWorldSecure">
    <input message="s0:HelloWorldSecureHttpPostIn"/>
    <output message="s0:HelloWorldSecureHttpPostOut"/>
  </operation>
</portType>
<binding name="HelloWorldSecureSoap" type="s0:HelloWorldSecureSoap">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
  <operation name="HelloWorldSecure">
    <soap:operation soapAction="http://workshop.bea.com/HelloWorldSecure/HelloWorldSecure">
      <input>
        <soap:body use="literal"/>
      </input>
      <output>
        <soap:body use="literal"/>
      </output>
    </soap:operation>
  </operation>
</binding>
<binding name="HelloWorldSecureHttpGet" type="s0:HelloWorldSecureHttpGet">
  <http:binding verb="GET"/>
  <operation name="HelloWorldSecure">
    <http:operation location="/HelloWorldSecure"/>
    <input>
      <http:urlEncoded/>
    </input>
    <output>
      <mime:mimeType part="Body"/>
    </output>
  </operation>
</binding>
<binding name="HelloWorldSecureHttpPost" type="s0:HelloWorldSecureHttpPost">
  <http:binding verb="POST"/>
  <operation name="HelloWorldSecure">
    <http:operation location="/HelloWorldSecure"/>
    <input>
      <mime:contentType type="application/x-www-form-urlencoded"/>
    </input>
    <output>
      <mime:mimeType part="Body"/>
    </output>
  </operation>
</binding>
<service name="HelloWorldSecure">
  <port name="HelloWorldSecureSoap" binding="s0:HelloWorldSecureSoap">

```

navWebServices.html Sample

```
<soap:address location="http://localhost:7001/WebServices/security/transport/helloWor
</port>
<port name="HelloWorldSecureHttpGet" binding="s0:HelloWorldSecureHttpGet">
  <http:address location="http://localhost:7001/WebServices/security/transport/helloWor
</port>
<port name="HelloWorldSecureHttpPost" binding="s0:HelloWorldSecureHttpPost">
  <http:address location="http://localhost:7001/WebServices/security/transport/helloWor
</port>
</service>
</definitions>
* ::
* /
```

wsse Samples

This section contains source code for the following samples.

Samples Included in This Section

callback Samples

reqResp Samples

usertoken Samples

callback Samples

This section contains source code for the following samples.

Samples Included in This Section

client Samples

Readme.html Sample

target Samples

client Samples

This section contains source code for the following samples.

Samples Included in This Section

Client.jws Sample

TargetControl.jcx Sample

TargetControlPolicy.wsse Sample

Client.jws Sample

This topic includes the source code for the Client.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/security/wsse/callback/client/

Sample Source Code

```
package security.wsse.callback.client;

/**
 * @common:target-namespace namespace="http://workshop.bea.com/Client"
 */
public class Client implements com.bea.jws.WebService
{
    public Callback callback;

    /**
     * @common:control
     */
    private security.wsse.callback.client.TargetControl targetControl;

    /**
     * @common:operation
     * @common:message-buffer enable="true"
     * @jws:conversation phase="start"
     */
    public void invokeHello()
    {
        targetControl.hello();
    }

    public interface Callback extends com.bea.control.ServiceControl
    {
        /**
         * @jws:conversation phase="finish"
         * @common:message-buffer enable="true"
         */
        void callback(java.lang.String message);
    }

    public void targetControl_callback(java.lang.String message)
    {
        callback.callback(message);
    }
}
```

TargetControl.jcx Sample

This topic includes the source code for the TargetControl.jcx Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/security/wsse/callback/client/

Sample Source Code

```
package security.wsse.callback.client;

/**
 * @jc:location http-url="http://localhost:7001/WebServices/security/wsse/callback/target/TargetControl.jcx"
 * @jc:wSDL file="#TargetWSDL"
 * @jc:ws-security-service file="TargetControlPolicy.wsse"
 * @jc:ws-security-callback file="TargetControlPolicy.wsse"
 */
public interface TargetControl extends com.bea.control.ControlExtension, com.bea.control.ServiceControl {
    public static class StartHeader
        implements java.io.Serializable
    {
        public java.lang.String conversationID;
        public java.lang.String callbackLocation;
    }

    public static class ContinueHeader
        implements java.io.Serializable
    {
        public java.lang.String conversationID;
    }

    public static class CallbackHeader
        implements java.io.Serializable
    {
        public java.lang.String conversationID;
    }

    public interface Callback extends com.bea.control.ServiceControl.Callback {
        /**
         * @jc:conversation phase="finish"
         */
        public void callback (java.lang.String message);
    }

    /**
     * @jc:conversation phase="start"
     */
    public void hello ();

    static final long serialVersionUID = 1L;
}
```

```

}

/** @common:define name="TargetWsd1" value::
  <?xml version="1.0" encoding="utf-8"?>
  <!-- @editor-info:link autogen="true" source="Target.jws" -->
  <definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:conv="http://www.openuri.org/2002/0
    <types>
      <s:schema elementFormDefault="qualified" targetNamespace="http://workshop.bea.com/TargetWsd1">
        <s:element name="hello">
          <s:complexType>
            <s:sequence/>
          </s:complexType>
        </s:element>
        <s:element name="helloResponse">
          <s:complexType>
            <s:sequence/>
          </s:complexType>
        </s:element>
        <s:element name="callbackResponse">
          <s:complexType>
            <s:sequence/>
          </s:complexType>
        </s:element>
        <s:element name="callback">
          <s:complexType>
            <s:sequence>
              <s:element name="message" type="s:string" minOccurs="0"/>
            </s:sequence>
          </s:complexType>
        </s:element>
      </s:schema>

      <s:schema elementFormDefault="qualified" targetNamespace="http://www.openuri.org/2002/0
        <s:element name="StartHeader" type="conv:StartHeader"/>
        <s:element name="ContinueHeader" type="conv:ContinueHeader"/>
        <s:element name="CallbackHeader" type="conv:CallbackHeader"/>
        <s:complexType name="StartHeader">
          <s:sequence>
            <s:element minOccurs="0" maxOccurs="1" name="conversationID" type="s:string"/>
            <s:element minOccurs="0" maxOccurs="1" name="callbackLocation" type="s:string"/>
          </s:sequence>
        </s:complexType>
        <s:complexType name="ContinueHeader">
          <s:sequence>
            <s:element minOccurs="1" maxOccurs="1" name="conversationID" type="s:string"/>
          </s:sequence>
        </s:complexType>
        <s:complexType name="CallbackHeader">
          <s:sequence>
            <s:element minOccurs="1" maxOccurs="1" name="conversationID" type="s:string"/>
          </s:sequence>
        </s:complexType>
      </s:schema>
    </types>
    <message name="helloSoapIn">
      <part name="parameters" element="s0:hello"/>
    </message>
    <message name="helloSoapOut">
      <part name="parameters" element="s0:helloResponse"/>
    </message>
    <message name="callbackSoapIn">

```

navWebServices.html Sample

```

    <part name="parameters" element="s0:callbackResponse"/>
</message>
<message name="callbackSoapOut">
    <part name="parameters" element="s0:callback"/>
</message>
<message name="helloHttpGetIn"/>
<message name="helloHttpGetOut"/>
<message name="callbackHttpGetIn"/>
<message name="callbackHttpGetOut">
    <part name="message" type="s:string"/>
</message>
<message name="helloHttpPostIn"/>
<message name="helloHttpPostOut"/>
<message name="callbackHttpPostIn"/>
<message name="callbackHttpPostOut">
    <part name="message" type="s:string"/>
</message>
<message name="StartHeader_literal">
    <part name="StartHeader" element="conv:StartHeader"/>
</message>
<message name="CallbackHeader_literal">
    <part name="CallbackHeader" element="conv:CallbackHeader"/>
</message>
<portType name="TargetSoap">
    <operation name="hello">
        <input message="s0:helloSoapIn"/>
        <output message="s0:helloSoapOut"/>
    </operation>
    <operation name="callback">
        <output message="s0:callbackSoapOut"/>
        <input message="s0:callbackSoapIn"/>
    </operation>
</portType>
<portType name="TargetHttpGet">
    <operation name="hello">
        <input message="s0:helloHttpGetIn"/>
        <output message="s0:helloHttpGetOut"/>
    </operation>
    <operation name="callback">
        <output message="s0:callbackHttpGetOut"/>
        <input message="s0:callbackHttpGetIn"/>
    </operation>
</portType>
<portType name="TargetHttpPost">
    <operation name="hello">
        <input message="s0:helloHttpPostIn"/>
        <output message="s0:helloHttpPostOut"/>
    </operation>
    <operation name="callback">
        <output message="s0:callbackHttpPostOut"/>
        <input message="s0:callbackHttpPostIn"/>
    </operation>
</portType>
<binding name="TargetSoap" type="s0:TargetSoap">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
    <operation name="hello">
        <soap:operation soapAction="http://workshop.bea.com/Target/hello" style="document"/>
        <cw:transition phase="start"/>
        <input>
            <soap:body use="literal"/>
            <soap:header wsdl:required="true" message="s0:StartHeader_literal" part="StartHeader"/>
        </input>
    </operation>

```

navWebServices.html Sample

```

    </input>
    <output>
      <soap:body use="literal"/>
    </output>
  </operation>
  <operation name="callback">
    <soap:operation soapAction="http://workshop.bea.com/Target/callback" style="document" />
    <cw:transition phase="finish"/>
    <input>
      <soap:body use="literal"/>
    </input>
    <output>
      <soap:body use="literal"/>
      <soap:header wsdl:required="true" message="s0:CallbackHeader_literal" part="CallbackHeader" />
    </output>
  </operation>
</binding>
<binding name="TargetHttpGet" type="s0:TargetHttpGet">
  <http:binding verb="GET"/>
  <operation name="hello">
    <http:operation location="/hello"/>
    <cw:transition phase="start"/>
    <input>
      <http:urlEncoded/>
    </input>
    <output/>
  </operation>
  <operation name="callback">
    <http:operation location="/callback"/>
    <cw:transition phase="finish"/>
    <input>
      <mime:mimeType part="Body"/>
    </input>
    <output>
      <http:urlEncoded/>
    </output>
  </operation>
</binding>
<binding name="TargetHttpPost" type="s0:TargetHttpPost">
  <http:binding verb="POST"/>
  <operation name="hello">
    <http:operation location="/hello"/>
    <cw:transition phase="start"/>
    <input>
      <mime:content type="application/x-www-form-urlencoded"/>
    </input>
    <output/>
  </operation>
  <operation name="callback">
    <http:operation location="/callback"/>
    <cw:transition phase="finish"/>
    <input>
      <mime:mimeType part="Body"/>
    </input>
    <output>
      <mime:content type="application/x-www-form-urlencoded"/>
    </output>
  </operation>
</binding>
<service name="Target">
  <port name="TargetSoap" binding="s0:TargetSoap">

```

navWebServices.html Sample

```
    <soap:address location="http://localhost:7001/WebServices/security/wsse/callback/targ
</port>
<port name="TargetHttpGet" binding="s0:TargetHttpGet">
    <http:address location="http://localhost:7001/WebServices/security/wsse/callback/targ
</port>
<port name="TargetHttpPost" binding="s0:TargetHttpPost">
    <http:address location="http://localhost:7001/WebServices/security/wsse/callback/targ
</port>
</service>
</definitions>
* ::
* /
```

TargetControlPolicy.wsse Sample

This topic includes the source code for the TargetControlPolicy.wsse Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/security/wsse/callback/client/

Sample Source Code

```
<?xml version="1.0" ?>
<wsSecurityPolicy xsi:schemaLocation="WSSecurity-policy.xsd"
  xmlns="http://www.bea.com/2003/03/wsse/config"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <!--
  Incoming SOAP messages must be accompanied by a username and password.
  -->
    <wsSecurityIn>
      <token tokenType="username"/>
    </wsSecurityIn>

  <!--
  Accompany outgoing SOAP messages with a username and password before sending them
  out on the wire.
  -->
    <wsSecurityOut>
      <userNameToken>
        <userName>weblogic</userName>
        <password type="TEXT">weblogic</password>
      </userNameToken>
    </wsSecurityOut>

</wsSecurityPolicy>
```

Readme.html Sample

This topic includes the source code for the Readme.html Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/security/wsse/callback/

Sample Source Code

```
<html>
  <head>
  </head>
  <body>
    <h1>callback Sample</h1>

    <p>The reqResp sample demonstrates how WSSE processes the SOAP messages passed between
    two asynchronously communicating web services.

    <p>Four SOAP messages are processed in this case.

    <ol>
    <li>This SOAP message contains the initial invocation of Target.jws's hello() method.</li>
    <li>This SOAP message contains the void return value of Target.jws's hello() method.
    <pre>&lt;Void xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true">&lt;/Void
    </li>
    <li>This SOAP message contains the content of TargetControl.jcx's callback() method sent to
    <li>This SOAP message contains the callback's void return value</li>
    </ol>

    <p>The diagram below shows how these SOAP messages are processed by the WSSE
    files ClientPolicy.wsse and TargetPolicy.wsse.
    <p>
    </body>
</html>
```


target Samples

This section contains source code for the following samples.

Samples Included in This Section

Target.jws Sample

TargetPolicy.wsse Sample

Target.jws Sample

This topic includes the source code for the Target.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/security/wsse/callback/target/

Sample Source Code

```
package security.wsse.callback.target;

/**
 * @jws:ws-security-service file="TargetPolicy.wsse"
 * @jws:ws-security-callback file="TargetPolicy.wsse"
 * @common:target-namespace namespace="http://workshop.bea.com/Target"
 */
public class Target implements com.bea.jws.WebService
{
    public Callback callback;

    /**
     * @common:operation
     * @common:message-buffer enable="true"
     * @jws:conversation phase="start"
     */
    public void hello()
    {
        callback.callback("Hello, Client!");
    }

    public interface Callback extends com.bea.control.ServiceControl
    {
        /**
         * @common:message-buffer enable="true"
         * @jws:conversation phase="finish"
         */
        void callback(String message);
    }
}
```

TargetPolicy.wsse Sample

This topic includes the source code for the TargetPolicy.wsse Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/security/wsse/callback/target/

Sample Source Code

```
<?xml version="1.0" ?>
<wsSecurityPolicy xsi:schemaLocation="WSSecurity-policy.xsd"
  xmlns="http://www.bea.com/2003/03/wsse/config"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <!--
  Incoming SOAP messages must be accompanied by a username and password.
  -->
    <wsSecurityIn>
      <token tokenType="username"/>
    </wsSecurityIn>

  <!--
  Enhance outgoing SOAP messages with a username and password before
  sending them out over the wire.
  -->
    <wsSecurityOut>
      <userNameToken>
        <userName>weblogic</userName>
        <password type="TEXT">weblogic</password>
      </userNameToken>
    </wsSecurityOut>

</wsSecurityPolicy>
```

reqResp Samples

This section contains source code for the following samples.

Samples Included in This Section

client Samples

mycompany Samples

Readme.html Sample

client Samples

This section contains source code for the following samples.

Samples Included in This Section

Client.jws Sample

MyCompanyControl.jcx Sample

MyCompanyControlPolicy.wsse Sample

Client.jws Sample

This topic includes the source code for the Client.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/security/wsse/reqResp/client/

Sample Source Code

```
package security.wsse.reqResp.client;

/**
 * @common:target-namespace namespace="http://workshop.bea.com/Client"
 */
public class Client implements com.bea.jws.WebService
{
    /**
     * @common:control
     */
    private security.wsse.reqResp.client.MyCompanyControl myCompanyControl;

    /**
     * @common:operation
     */
    public String invokeHello()
    {
        return myCompanyControl.hello();
    }
}
```

MyCompanyControl.jcx Sample

This topic includes the source code for the MyCompanyControl.jcx Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/security/wsse/reqResp/client/

Sample Source Code

```
package security.wsse.reqResp.client;

/**
 * @jc:location http-url="http://localhost:7001/WebServices/security/wsse/reqResp/mycompany/MyCompanyControl.jcx"
 * @jc:wSDL file="#MyCompanyWSDL"
 * @jc:ws-security-service file="MyCompanyControlPolicy.wsse"
 */
public interface MyCompanyControl extends com.bea.control.ControlExtension, com.bea.control.Service {

    public java.lang.String hello ();

    static final long serialVersionUID = 1L;
}

/** @common:define name="MyCompanyWSDL" value::
<?xml version="1.0" encoding="utf-8"?>
<!-- @editor-info:link autogen="true" source="MyCompany.jws" -->
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:conv="http://www.openuri.org/2003/11/conv"
    <types>
        <s:schema elementFormDefault="qualified" targetNamespace="http://workshop.bea.com/MyCompanyControl"
            <s:element name="hello">
                <s:complexType>
                    <s:sequence/>
                </s:complexType>
            </s:element>
            <s:element name="helloResponse">
                <s:complexType>
                    <s:sequence>
                        <s:element name="helloResult" type="s:string" minOccurs="0"/>
                    </s:sequence>
                </s:complexType>
            </s:element>
            <s:element name="string" nillable="true" type="s:string"/>
        </s:schema>
    </types>
    <message name="helloSoapIn">
        <part name="parameters" element="s0:hello"/>
    </message>
    <message name="helloSoapOut">
        <part name="parameters" element="s0:helloResponse"/>
    </message>
    <message name="helloHttpGetIn"/>
    <message name="helloHttpGetOut"/>
</definitions>
```

navWebServices.html Sample

```

    <part name="Body" element="s0:string"/>
</message>
<message name="helloHttpPostIn"/>
<message name="helloHttpPostOut">
    <part name="Body" element="s0:string"/>
</message>
<portType name="MyCompanySoap">
    <operation name="hello">
        <input message="s0:helloSoapIn"/>
        <output message="s0:helloSoapOut"/>
    </operation>
</portType>
<portType name="MyCompanyHttpGet">
    <operation name="hello">
        <input message="s0:helloHttpGetIn"/>
        <output message="s0:helloHttpGetOut"/>
    </operation>
</portType>
<portType name="MyCompanyHttpPost">
    <operation name="hello">
        <input message="s0:helloHttpPostIn"/>
        <output message="s0:helloHttpPostOut"/>
    </operation>
</portType>
<binding name="MyCompanySoap" type="s0:MyCompanySoap">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
    <operation name="hello">
        <soap:operation soapAction="http://workshop.bea.com/MyCompany/hello" style="document">
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
        </soap:operation>
    </operation>
</binding>
<binding name="MyCompanyHttpGet" type="s0:MyCompanyHttpGet">
    <http:binding verb="GET"/>
    <operation name="hello">
        <http:operation location="/hello"/>
        <input>
            <http:urlEncoded/>
        </input>
        <output>
            <mime:mimeType part="Body"/>
        </output>
    </operation>
</binding>
<binding name="MyCompanyHttpPost" type="s0:MyCompanyHttpPost">
    <http:binding verb="POST"/>
    <operation name="hello">
        <http:operation location="/hello"/>
        <input>
            <mime:content type="application/x-www-form-urlencoded"/>
        </input>
        <output>
            <mime:mimeType part="Body"/>
        </output>
    </operation>
</binding>
<service name="MyCompany">

```


navWebServices.html Sample

```
<port name="MyCompanySoap" binding="s0:MyCompanySoap">
  <soap:address location="http://localhost:7001/WebServices/security/wsse/reqResp/mycom">
  </port>
<port name="MyCompanyHttpGet" binding="s0:MyCompanyHttpGet">
  <http:address location="http://localhost:7001/WebServices/security/wsse/reqResp/mycom">
  </port>
<port name="MyCompanyHttpPost" binding="s0:MyCompanyHttpPost">
  <http:address location="http://localhost:7001/WebServices/security/wsse/reqResp/mycom">
  </port>
</service>
</definitions>
* ::
*/
```

MyCompanyControlPolicy.wsse Sample

This topic includes the source code for the MyCompanyControlPolicy.wsse Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/security/wsse/reqResp/client/

Sample Source Code

```
<wsSecurityPolicy xsi:schemaLocation="WSSecurity-policy.xsd" xmlns="http://www.bea.com/2003/03/04/WSSecurityPolicy" >
  <wsSecurityOut>
    <!--
    Accompany the SOAP message with a valid username and password
    -->
    <userNameToken>
      <userName>weblogic</userName>
      <password type="TEXT">weblogic</password>
    </userNameToken>
    <!--
    Encrypt the SOAP message with the recipient's (MyCompany.jws) public key.
    Only the recipient's private key can decrypt it.
    Ensures the confidentiality of the SOAP message.
    (This process requires that the sender's keystore already contains
    a digital certificate containing the recipients public key.)
    -->
    <encryption>
      <encryptionKey>
        <alias>mycompany</alias>
      </encryptionKey>
    </encryption>
    <!--
    Sign the SOAP message with the sender's (Client.jws) private key.
    Only the sender's public key can validate the signature.
    Ensures the authenticity of the sender, i.e., that the sender is
    in fact the source of the SOAP message.
    -->
    <signatureKey>
      <alias>client1</alias>
      <password>password</password>
    </signatureKey>
  </wsSecurityOut>

  <wsSecurityIn>
    <!--
    Incoming SOAP message must be accompanied by a valid username
    and password.
    -->
    <token tokenType="username"/>
    <!--
    Incoming SOAP messages must be encrypted with client.jws's
    public key. The alias and password to access the client.jws's
    decrypting private key in the keystore are provided by
    the <decryptionKey> element below.
    -->
```

navWebServices.html Sample

```
<encryptionRequired>
  <decryptionKey>
    <alias>client1</alias>
    <password>password</password>
  </decryptionKey>
</encryptionRequired>

<!--
Incoming SOAP messages must be digitally signed with the sender's
private key.
The sender's public key is used to validate the signature.
-->
  <signatureRequired>true</signatureRequired>
</wsSecurityIn>

<!--
Look for the client.jks keystore in the default location, the server domain
root, in this case, BEA_HOME\weblogic81\samples\domains\workshop.
-->
  <keyStore>
    <keyStoreLocation>samples_client.jks</keyStoreLocation>
    <keyStorePassword>password</keyStorePassword>
  </keyStore>
</wsSecurityPolicy>
```

mycompany Samples

This section contains source code for the following samples.

Samples Included in This Section

MyCompany.jws Sample

MyCompanyContract.wsdl Sample

MyCompanySecurityPolicy.wsse Sample

MyCompany.jws Sample

This topic includes the source code for the MyCompany.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/security/wsse/reqResp/mycompany/

Sample Source Code

```
package security.wsse.reqResp.mycompany;

/**
 * @jws:ws-security-service file="MyCompanySecurityPolicy.wsse"
 * @common:target-namespace namespace="http://workshop.bea.com/MyCompany"
 */
public class MyCompany implements com.bea.jws.WebService
{

    /**
     * @common:operation
     */
    public String hello()
    {
        return "Hello, Client!";
    }
}
```

MyCompanyContract.wsdl Sample

This topic includes the source code for the MyCompanyContract.wsdl Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/security/wsse/reqResp/mycompany/

Sample Source Code

```
<?xml version="1.0" encoding="utf-8"?>
<!-- @editor-info:link autogen="true" source="MyCompany.jws" -->
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:conv="http://www.openuri.org/2002/01/
  <types>
    <s:schema elementFormDefault="qualified" targetNamespace="http://workshop.bea.com/MyCompany">
      <s:element name="hello">
        <s:complexType>
          <s:sequence/>
        </s:complexType>
      </s:element>
      <s:element name="helloResponse">
        <s:complexType>
          <s:sequence>
            <s:element name="helloResult" type="s:string" minOccurs="0"/>
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="string" nillable="true" type="s:string"/>
    </s:schema>
  </types>
  <message name="helloSoapIn">
    <part name="parameters" element="s0:hello"/>
  </message>
  <message name="helloSoapOut">
    <part name="parameters" element="s0:helloResponse"/>
  </message>
  <message name="helloHttpGetIn"/>
  <message name="helloHttpGetOut">
    <part name="Body" element="s0:string"/>
  </message>
  <message name="helloHttpPostIn"/>
  <message name="helloHttpPostOut">
    <part name="Body" element="s0:string"/>
  </message>
  <portType name="MyCompanySoap">
    <operation name="hello">
      <input message="s0:helloSoapIn"/>
      <output message="s0:helloSoapOut"/>
    </operation>
  </portType>
  <portType name="MyCompanyHttpGet">
    <operation name="hello">
      <input message="s0:helloHttpGetIn"/>
      <output message="s0:helloHttpGetOut"/>
    </operation>
  </portType>
```

navWebServices.html Sample

```

    </operation>
</portType>
<portType name="MyCompanyHttpPost">
    <operation name="hello">
        <input message="s0:helloHttpPostIn"/>
        <output message="s0:helloHttpPostOut"/>
    </operation>
</portType>
<binding name="MyCompanySoap" type="s0:MyCompanySoap">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
    <operation name="hello">
        <soap:operation soapAction="http://workshop.bea.com/MyCompany/hello" style="document"/>
        <input>
            <soap:body use="literal"/>
        </input>
        <output>
            <soap:body use="literal"/>
        </output>
    </operation>
</binding>
<binding name="MyCompanyHttpGet" type="s0:MyCompanyHttpGet">
    <http:binding verb="GET"/>
    <operation name="hello">
        <http:operation location="/hello"/>
        <input>
            <http:urlEncoded/>
        </input>
        <output>
            <mime:mimeXml part="Body"/>
        </output>
    </operation>
</binding>
<binding name="MyCompanyHttpPost" type="s0:MyCompanyHttpPost">
    <http:binding verb="POST"/>
    <operation name="hello">
        <http:operation location="/hello"/>
        <input>
            <mime:content type="application/x-www-form-urlencoded"/>
        </input>
        <output>
            <mime:mimeXml part="Body"/>
        </output>
    </operation>
</binding>
<service name="MyCompany">
    <port name="MyCompanySoap" binding="s0:MyCompanySoap">
        <soap:address location="http://localhost:7001/WebServices/security/wsse/reqResp/mycompany">
        </soap:address>
    </port>
    <port name="MyCompanyHttpGet" binding="s0:MyCompanyHttpGet">
        <http:address location="http://localhost:7001/WebServices/security/wsse/reqResp/mycompany">
        </http:address>
    </port>
    <port name="MyCompanyHttpPost" binding="s0:MyCompanyHttpPost">
        <http:address location="http://localhost:7001/WebServices/security/wsse/reqResp/mycompany">
        </http:address>
    </port>
</service>
</definitions>

```

MyCompanySecurityPolicy.wsse Sample

This topic includes the source code for the MyCompanySecurityPolicy.wsse Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/security/wsse/reqResp/mycompany/

Sample Source Code

```
<wsSecurityPolicy xsi:schemaLocation="WSSecurity-policy.xsd" xmlns="http://www.bea.com/2003/03/
```

```
    <wsSecurityIn>
      <!--
      Incoming SOAP message must be accompanied by a valid username
      and password.
      -->
        <token tokenType="username"/>
      <!--
      Incoming SOAP messages must be encrypted with mycompany.jws's
      public key. The alias and password to access the mycompany.jws's
      decrypting private key in the keystore are provided by
      the <decryptionKey> element below.
      -->
        <encryptionRequired>
          <decryptionKey>
            <alias>mycompany</alias>
            <password>password</password>
          </decryptionKey>
        </encryptionRequired>
      <!--
      Incoming SOAP messages must be digitally signed with the sender's
      private key.
      The sender's public key is used to validate the signature.
      -->
        <signatureRequired>true</signatureRequired>
    </wsSecurityIn>

    <wsSecurityOut>
      <!--
      Accompany the SOAP message with a valid username and password
      -->
      <userNameToken>
        <userName>weblogic</userName>
        <password type="TEXT">weblogic</password>
      </userNameToken>
      <!--
      Encrypt the SOAP message with the recipient's (Client.jws) public key.
      Only the recipient's private key can decrypt it.
      Ensures the confidentiality of the SOAP message.
      (This process requires that the sender's keystore already contains
      a digital certificate containing the recipients public key.)
      -->
        <encryption>
          <encryptionKey>
```


navWebServices.html Sample

```
        <alias>client1</alias>
      </encryptionKey>
    </encryption>
    <!--
    Sign the SOAP message with the sender's (MyCompany.jws) private key.
    Only the sender's public key can validate the signature.
    Ensures the authenticity of the sender, i.e., that the sender is
    in fact the source of the SOAP message.
    -->
    <signatureKey>
      <alias>mycompany</alias>
      <password>password</password>
    </signatureKey>
  </wsSecurityOut>

  <!--
  Look for the mycompany.jks keystore in the default location, the server domain
  root, in this case, BEA_HOME\weblogic81\samples\domains\workshop.
  -->
  <keyStore>
    <keyStoreLocation>samples_mycompany.jks</keyStoreLocation>
    <keyStorePassword>password</keyStorePassword>
  </keyStore>
</wsSecurityPolicy>
```

Readme.html Sample

This topic includes the source code for the Readme.html Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/security/wsse/reqResp/

Sample Source Code

```
<html>
  <head>
  </head>
  <body>
    <h1>reqResp Sample</h1>

    <p>The reqResp sample demonstrates synchronous communication between two web services
    where both web services require that incoming SOAP messages
    be accompanied by a username/password, a digital certificate, and encryption.

    <p>
    The figure below illustrates how WSSE processes SOAP messages in a
    request/response cycle, where Client.jws initiates a request.

    <p>
  </body>
</html>
```

usertoken Samples

This section contains source code for the following samples.

Samples Included in This Section

Readme.html Sample

webServiceA Samples

webServiceB Samples

Readme.html Sample

This topic includes the source code for the Readme.html Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/security/wsse/usertoken/

Sample Source Code

```
<html>
  <head>
  </head>
  <body>
    <h1>userToken Sample</h1>

    <p>The userToken sample shows a simple case of synchronous web service communication, where
    both web service require username and password.

    <blockquote><b>Note</b> that WebServiceB.jws is also used to demonstrate the use of Java pr
    classes. For sample code demonstrating the use of Java proxy client classes, see
    SamplesApp/ProxyClient/WSSE/.</blockquote>

    <p>Web service A sends a SOAP message to invoke web service B's hello() method.
    Web service B responds by sending a SOAP message containing the hello() method's
    return value.

    <p>Both web services require that incoming SOAP messages be accompanied by a
    valued user name and password and both web services wrap outgoing SOAP messages
    with a user name and password before they are sent over the wire.

    <p>
    The diagram below shows how the invoking SOAP message and the return SOAP
    message are processed by WSSE.

    <p></body>
</html>
```

webServiceA Samples

This section contains source code for the following samples.

Samples Included in This Section

WebServiceA.jws Sample

WebServiceBControl.jcx Sample

WebServiceBControlPolicy.wsse Sample

WebServiceA.jws Sample

This topic includes the source code for the WebServiceA.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/security/wsse/usertoken/webServiceA/

Sample Source Code

```
package security.wsse.usertoken.webServiceA;

/**
 * @common:target-namespace namespace="http://workshop.bea.com/WebServiceA"
 */
public class WebServiceA implements com.bea.jws.WebService
{
    /**
     * @common:control
     */
    private security.wsse.usertoken.webServiceA.WebServiceBControl webServiceBControl;

    /**
     * @common:operation
     */
    public String invokeHello()
    {
        return webServiceBControl.hello();
    }
}
```

WebServiceBControl.jcx Sample

This topic includes the source code for the WebServiceBControl.jcx Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/security/wsse/usertoken/webServiceA/

Sample Source Code

```
package security.wsse.usertoken.webServiceA;

/**
 * @jc:location http-url="http://localhost:7001/WebServices/security/wsse/usertoken/webServiceB"
 * @jc:wSDL file="#WebServiceBWSDL"
 * @jc:ws-security-service file="WebServiceBControlPolicy.wsse"
 */
public interface WebServiceBControl extends com.bea.control.ControlExtension, com.bea.control.S
{
    public java.lang.String hello ();

    static final long serialVersionUID = 1L;
}

/** @common:define name="WebServiceBWSDL" value::
<?xml version="1.0" encoding="utf-8"?>
<!-- @editor-info:link autogen="true" source="WebServiceB.jws" -->
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:conv="http://www.openuri.org/20
    <types>
        <s:schema elementFormDefault="qualified" targetNamespace="http://workshop.bea.com/WebSe
            <s:element name="hello">
                <s:complexType>
                    <s:sequence/>
                </s:complexType>
            </s:element>
            <s:element name="helloResponse">
                <s:complexType>
                    <s:sequence>
                        <s:element name="helloResult" type="s:string" minOccurs="0"/>
                    </s:sequence>
                </s:complexType>
            </s:element>
            <s:element name="string" nillable="true" type="s:string"/>
        </s:schema>

    </types>
    <message name="helloSoapIn">
        <part name="parameters" element="s0:hello"/>
    </message>
    <message name="helloSoapOut">
        <part name="parameters" element="s0:helloResponse"/>
    </message>
    <message name="helloHttpGetIn"/>
    <message name="helloHttpGetOut">
```

navWebServices.html Sample

```

    <part name="Body" element="s0:string"/>
</message>
<message name="helloHttpPostIn"/>
<message name="helloHttpPostOut">
    <part name="Body" element="s0:string"/>
</message>
<portType name="WebServiceBSoap">
    <operation name="hello">
        <input message="s0:helloSoapIn"/>
        <output message="s0:helloSoapOut"/>
    </operation>
</portType>
<portType name="WebServiceBHttpGet">
    <operation name="hello">
        <input message="s0:helloHttpGetIn"/>
        <output message="s0:helloHttpGetOut"/>
    </operation>
</portType>
<portType name="WebServiceBHttpPost">
    <operation name="hello">
        <input message="s0:helloHttpPostIn"/>
        <output message="s0:helloHttpPostOut"/>
    </operation>
</portType>
<binding name="WebServiceBSoap" type="s0:WebServiceBSoap">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
    <operation name="hello">
        <soap:operation soapAction="http://workshop.bea.com/WebServiceB/hello" style="document"/>
        <input>
            <soap:body use="literal"/>
        </input>
        <output>
            <soap:body use="literal"/>
        </output>
    </operation>
</binding>
<binding name="WebServiceBHttpGet" type="s0:WebServiceBHttpGet">
    <http:binding verb="GET"/>
    <operation name="hello">
        <http:operation location="/hello"/>
        <input>
            <http:urlEncoded/>
        </input>
        <output>
            <mime:mimeType part="Body"/>
        </output>
    </operation>
</binding>
<binding name="WebServiceBHttpPost" type="s0:WebServiceBHttpPost">
    <http:binding verb="POST"/>
    <operation name="hello">
        <http:operation location="/hello"/>
        <input>
            <mime:contentType type="application/x-www-form-urlencoded"/>
        </input>
        <output>
            <mime:mimeType part="Body"/>
        </output>
    </operation>
</binding>
<service name="WebServiceB">

```


navWebServices.html Sample

```
<port name="WebServiceBSoap" binding="s0:WebServiceBSoap">
  <soap:address location="http://localhost:7001/WebServices/security/wsse/usertoken/web">
  </port>
<port name="WebServiceBHttpGet" binding="s0:WebServiceBHttpGet">
  <http:address location="http://localhost:7001/WebServices/security/wsse/usertoken/web">
  </port>
<port name="WebServiceBHttpPost" binding="s0:WebServiceBHttpPost">
  <http:address location="http://localhost:7001/WebServices/security/wsse/usertoken/web">
  </port>
</service>
</definitions>
* ::
*/
```

WebServiceBControlPolicy.wsse Sample

This topic includes the source code for the WebServiceBControlPolicy.wsse Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/security/wsse/usertoken/webServiceA/

Sample Source Code

```
<?xml version="1.0" ?>
<wsSecurityPolicy xsi:schemaLocation="WSSecurity-policy.xsd"
  xmlns="http://www.bea.com/2003/03/wsse/config"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <!--
  Incoming SOAP messages must be accompanied by a username and password.
  -->
    <wsSecurityIn>
      <token tokenType="username"/>
    </wsSecurityIn>

  <!--
  Accompany outgoing SOAP messages with a username and password before sending them
  out on the wire.
  -->
    <wsSecurityOut>
      <userNameToken>
        <userName>weblogic</userName>
        <password type="TEXT">weblogic</password>
      </userNameToken>
    </wsSecurityOut>

</wsSecurityPolicy>
```

webServiceB Samples

This section contains source code for the following samples.

Samples Included in This Section

PolicyB.wsse Sample

WebServiceB.jws Sample

WebServiceB.wsdl Sample

PolicyB.wsse Sample

This topic includes the source code for the PolicyB.wsse Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/security/wsse/usertoken/webServiceB/

Sample Source Code

```
<?xml version="1.0" ?>
<wsSecurityPolicy xsi:schemaLocation="WSecurity-policy.xsd"
  xmlns="http://www.bea.com/2003/03/wsse/config"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <!--
  Incoming SOAP messages must be accompanied by a username and password.
  -->
    <wsSecurityIn>
      <token tokenType="username"/>
    </wsSecurityIn>

  <!--
  Accompany outgoing SOAP messages with a username and password before sending them
  out on the wire.
  -->
    <wsSecurityOut>
      <userNameToken>
        <userName>weblogic</userName>
        <password type="TEXT">weblogic</password>
      </userNameToken>
    </wsSecurityOut>

</wsSecurityPolicy>
```

WebServiceB.jws Sample

This topic includes the source code for the WebServiceB.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/security/wsse/usertoken/webServiceB/

Sample Source Code

```
package security.wsse.usertoken.webServiceB;

/**
 * @jws:ws-security-service file="PolicyB.wsse"
 * @common:target-namespace namespace="http://workshop.bea.com/WebServiceB"
 */
public class WebServiceB implements com.bea.jws.WebService
{
    /**
     * @common:operation
     */
    public String hello()
    {
        return "Hello, WebServiceA!";
    }
}
```

WebServiceB.wsdl Sample

This topic includes the source code for the WebServiceB.wsdl Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/security/wsse/usertoken/webServiceB/

Sample Source Code

```
<?xml version="1.0" encoding="utf-8"?>
<!-- @editor-info:link autogen="true" source="WebServiceB.jws" -->
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:conv="http://www.openuri.org/2002/01/
  <types>
    <s:schema elementFormDefault="qualified" targetNamespace="http://workshop.bea.com/WebServiceB">
      <s:element name="hello">
        <s:complexType>
          <s:sequence/>
        </s:complexType>
      </s:element>
      <s:element name="helloResponse">
        <s:complexType>
          <s:sequence>
            <s:element name="helloResult" type="s:string" minOccurs="0"/>
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="string" nillable="true" type="s:string"/>
    </s:schema>
  </types>
  <message name="helloSoapIn">
    <part name="parameters" element="s0:hello"/>
  </message>
  <message name="helloSoapOut">
    <part name="parameters" element="s0:helloResponse"/>
  </message>
  <message name="helloHttpGetIn"/>
  <message name="helloHttpGetOut">
    <part name="Body" element="s0:string"/>
  </message>
  <message name="helloHttpPostIn"/>
  <message name="helloHttpPostOut">
    <part name="Body" element="s0:string"/>
  </message>
  <portType name="WebServiceBSoap">
    <operation name="hello">
      <input message="s0:helloSoapIn"/>
      <output message="s0:helloSoapOut"/>
    </operation>
  </portType>
  <portType name="WebServiceBHttpGet">
    <operation name="hello">
      <input message="s0:helloHttpGetIn"/>
      <output message="s0:helloHttpGetOut"/>
    </operation>
  </portType>
```

navWebServices.html Sample

```

    </operation>
</portType>
<portType name="WebServiceBHttpPost">
    <operation name="hello">
        <input message="s0:helloHttpPostIn"/>
        <output message="s0:helloHttpPostOut"/>
    </operation>
</portType>
<binding name="WebServiceBSoap" type="s0:WebServiceBSoap">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
    <operation name="hello">
        <soap:operation soapAction="http://workshop.bea.com/WebServiceB/hello" style="document"/>
        <input>
            <soap:body use="literal"/>
        </input>
        <output>
            <soap:body use="literal"/>
        </output>
    </operation>
</binding>
<binding name="WebServiceBHttpGet" type="s0:WebServiceBHttpGet">
    <http:binding verb="GET"/>
    <operation name="hello">
        <http:operation location="/hello"/>
        <input>
            <http:urlEncoded/>
        </input>
        <output>
            <mime:mimeType part="Body"/>
        </output>
    </operation>
</binding>
<binding name="WebServiceBHttpPost" type="s0:WebServiceBHttpPost">
    <http:binding verb="POST"/>
    <operation name="hello">
        <http:operation location="/hello"/>
        <input>
            <mime:contentType="application/x-www-form-urlencoded"/>
        </input>
        <output>
            <mime:mimeType part="Body"/>
        </output>
    </operation>
</binding>
<service name="WebServiceB">
    <port name="WebServiceBSoap" binding="s0:WebServiceBSoap">
        <soap:address location="http://localhost:7001/WebServices/security/wsse/usertoken/webServ
    </port>
    <port name="WebServiceBHttpGet" binding="s0:WebServiceBHttpGet">
        <http:address location="http://localhost:7001/WebServices/security/wsse/usertoken/webServ
    </port>
    <port name="WebServiceBHttpPost" binding="s0:WebServiceBHttpPost">
        <http:address location="http://localhost:7001/WebServices/security/wsse/usertoken/webServ
    </port>
</service>
</definitions>

```

service Samples

This section contains source code for the following samples.

Samples Included in This Section

QuoteClient.jws Sample

QuoteService.jws Sample

QuoteServiceControl.jcx Sample

QuoteClient.jws Sample

This topic includes the source code for the QuoteClient.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/service/

Sample Source Code

```
package service;

/**
 * <p>Demonstrates modification of a Service control's JCX file to change
 * the way a web service looks to its clients. This web service is merely
 * a test harness for QuoteServiceControl.jcx, where the technique of
 * modifying a JCX file is demonstrated. See the comments in
 * QuoteServiceControl.jcx for an explanation of this sample.</p>
 */
 * @common:target-namespace namespace="http://workshop.bea.com/QuoteClient"
 */
public class QuoteClient implements com.bea.jws.WebService
{
    /**
     * @common:control
     */
    private service.QuoteServiceControl quoteServiceControl;

    public Callback callback;

    public interface Callback
    {
        /**
         * @jws:conversation phase="finish"
         */
        public void onReply(double quote);
    }

    /**
     * @common:operation
     * @jws:conversation phase="start"
     */
    public void getQuote(String tickerSymbol)
    {
        quoteServiceControl.getQuote(tickerSymbol);
    }

    private void quoteServiceControl_onQuoteReady(java.lang.String tickerSymbol, double dQuote)
    {
        callback.onReply(dQuote);
    }
}
```

QuoteService.jws Sample

This topic includes the source code for the QuoteService.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/service/

Sample Source Code

```
package service;

import java.lang.Math;
import com.bea.control.TimerControl;

/**
 * <p>This is the original service from which QuoteServiceControl.jcx was
 * generated. Notice that the getQuote method takes two parameters. In
 * QuoteService.jcx, the signature of the method is changed to accept a
 * single parameter and hard-code the other.</p>
 *
 * <p>See the comments in QuoteServiceControl.jcx for details.</p>
 *
 * @common:target-namespace namespace="http://workshop.bea.com/QuoteService"
 */
public class QuoteService implements com.bea.jws.WebService
{
    String symbol;

    /**
     * @common:control
     * @jws:timer timeout="5 seconds"
     */
    private TimerControl timer;

    public Callback callback;

    public interface Callback
    {
        /**
         * @jws:conversation phase="finish"
         */
        public void onQuoteReady(String tickerSymbol, double dQuote);
    }

    /**
     * This method is called by the QuoteClient.jws web service
     * via the QuoteServiceControl control. Note that although the
     * client isn't passing a customer ID value, this service receives
     * one because the value is hardcoded in an XQuery map on the
     * service control. The customer ID is not used, but is printed
     * to the WebLogic Server console window.
     *
     * @common:operation
     * @jws:conversation phase="start"
     */
}
```

navWebServices.html Sample

```
    */
    public void getQuote(int customerID, String tickerSymbol)
    {
        System.out.println("customer ID: " + customerID);

        symbol = tickerSymbol;
        timer.start();
    }

    private void timer_onTimeout(long time)
    {
        callback.onQuoteReady( symbol, (double)(Math.random() * 30.0) );
    }
}
```

QuoteServiceControl.jcx Sample

This topic includes the source code for the QuoteServiceControl.jcx Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/service/

Sample Source Code

```
package service;

/**
 * <p>This Service control has been manually modified and is not
 * automatically generated from QuoteService.jws, although it still
 * calls QuoteService.jws.</p>
 *
 * <p>The getQuote method has a modified signature, with the customerID
 * parameter removed. The customerID parameter is hardcoded in an
 * XQuery map on the getQuote method. So the public contract (the format
 * of the XML message sent to the web service) is still the same, taking
 * two parameters, but the Java method a client of this control uses
 * takes only a single parameter. This is an example of the type
 * of modifications one can make to a Service control.</p>
 *
 * @jc:location http-url="QuoteService.jws" jms-url="QuoteService.jws"
 * @jc:wSDL file="#QuoteServiceWSDL"
 * @common:xmlns namespace="http://openuri.org/bea/samples/workshop/service/quoteService" prefix=""
 */
public interface QuoteServiceControl extends com.bea.control.ControlExtension, com.bea.control.Control {

    public static class StartHeader
        implements java.io.Serializable
    {
        public java.lang.String conversationID;
        public java.lang.String callbackLocation;
    }

    public static class ContinueHeader
        implements java.io.Serializable
    {
        public java.lang.String conversationID;
    }

    public static class CallbackHeader
        implements java.io.Serializable
    {
        public java.lang.String conversationID;
    }

    public interface Callback extends com.bea.control.ServiceControl.Callback {
        /**
         * @jc:conversation phase="finish"
         */
    }
}
```

navWebServices.html Sample

```

    */
    public void onQuoteReady (java.lang.String tickerSymbol, double dQuote);
}

/**
 * @jc:conversation phase="start"
 * @jc:parameter-xml schema-element="ns0:getQuote" xquery::
 * declare namespace ns1 = "http://workshop.bea.com/QuoteService"
 * declare namespace ns0 = "http://openuri.org/bea/samples/workshop/service/quoteService"
 *
 * <ns1:getQuote>
 *   <ns1:customerID>123456789</ns1:customerID>
 *   <ns1:tickerSymbol>{ data($input/ns0:tickerSymbol) }</ns1:tickerSymbol>
 * </ns1:getQuote>::
 */
public void getQuote (java.lang.String tickerSymbol);

static final long serialVersionUID = 1L;
}

/** @common:define name="QuoteServiceWsd1" value::
<?xml version="1.0" encoding="utf-8"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:conv="http://www.openuri.org/2002/01/11/quoteService"
  <types>
    <s:schema elementFormDefault="qualified" targetNamespace="http://workshop.bea.com/QuoteService">
      <s:element name="onQuoteReadyResponse">
        <s:complexType>
          <s:sequence/>
        </s:complexType>
      </s:element>
      <s:element name="onQuoteReady">
        <s:complexType>
          <s:sequence>
            <s:element name="tickerSymbol" type="s:string" minOccurs="0"/>
            <s:element name="dQuote" type="s:double"/>
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="getQuote">
        <s:complexType>
          <s:sequence>
            <s:element name="tickerSymbol" type="s:string" minOccurs="0"/>
          </s:sequence>
        </s:complexType>
      </s:element>
      <s:element name="getQuoteResponse">
        <s:complexType>
          <s:sequence/>
        </s:complexType>
      </s:element>
    </s:schema>

    <s:schema elementFormDefault="qualified" targetNamespace="http://www.openuri.org/2002/01/11/quoteService">
      <s:element name="StartHeader" type="conv:StartHeader"/>
      <s:element name="ContinueHeader" type="conv:ContinueHeader"/>
      <s:element name="CallbackHeader" type="conv:CallbackHeader"/>
      <s:complexType name="StartHeader">
        <s:sequence>
          <s:element minOccurs="0" maxOccurs="1" name="conversationID" type="s:string"/>
          <s:element minOccurs="0" maxOccurs="1" name="callbackLocation" type="s:string"/>
        </s:sequence>
      </s:complexType>
    </s:schema>
  </types>
</definitions>

```

navWebServices.html Sample

```

        </s:sequence>
    </s:complexType>
    <s:complexType name="ContinueHeader">
        <s:sequence>
            <s:element minOccurs="1" maxOccurs="1" name="conversationID" type="s:string"/>
        </s:sequence>
    </s:complexType>
    <s:complexType name="CallbackHeader">
        <s:sequence>
            <s:element minOccurs="1" maxOccurs="1" name="conversationID" type="s:string"/>
        </s:sequence>
    </s:complexType>
</s:schema>
</types>
<message name="onQuoteReadySoapIn">
    <part name="parameters" element="s0:onQuoteReadyResponse"/>
</message>
<message name="onQuoteReadySoapOut">
    <part name="parameters" element="s0:onQuoteReady"/>
</message>
<message name="getQuoteSoapIn">
    <part name="parameters" element="s0:getQuote"/>
</message>
<message name="getQuoteSoapOut">
    <part name="parameters" element="s0:getQuoteResponse"/>
</message>
<message name="onQuoteReadyHttpGetIn"/>
<message name="onQuoteReadyHttpGetOut">
    <part name="tickerSymbol" type="s:string"/>
    <part name="dQuote" type="s:string"/>
</message>
<message name="getQuoteHttpGetIn">
    <part name="tickerSymbol" type="s:string"/>
</message>
<message name="getQuoteHttpGetOut"/>
<message name="onQuoteReadyHttpPostIn"/>
<message name="onQuoteReadyHttpPostOut">
    <part name="tickerSymbol" type="s:string"/>
    <part name="dQuote" type="s:string"/>
</message>
<message name="getQuoteHttpPostIn">
    <part name="tickerSymbol" type="s:string"/>
</message>
<message name="getQuoteHttpPostOut"/>
<message name="StartHeader_literal">
    <part name="StartHeader" element="conv:StartHeader"/>
</message>
<message name="CallbackHeader_literal">
    <part name="CallbackHeader" element="conv:CallbackHeader"/>
</message>
<portType name="QuoteServiceSoap">
    <operation name="onQuoteReady">
        <output message="s0:onQuoteReadySoapOut"/>
        <input message="s0:onQuoteReadySoapIn"/>
    </operation>
    <operation name="getQuote">
        <input message="s0:getQuoteSoapIn"/>
        <output message="s0:getQuoteSoapOut"/>
    </operation>
</portType>
<portType name="QuoteServiceHttpGet">

```

navWebServices.html Sample

```

<operation name="onQuoteReady">
  <output message="s0:onQuoteReadyHttpGetOut" />
  <input message="s0:onQuoteReadyHttpGetIn" />
</operation>
<operation name="getQuote">
  <input message="s0:getQuoteHttpGetIn" />
  <output message="s0:getQuoteHttpGetOut" />
</operation>
</portType>
<portType name="QuoteServiceHttpPost">
  <operation name="onQuoteReady">
    <output message="s0:onQuoteReadyHttpPostOut" />
    <input message="s0:onQuoteReadyHttpPostIn" />
  </operation>
  <operation name="getQuote">
    <input message="s0:getQuoteHttpPostIn" />
    <output message="s0:getQuoteHttpPostOut" />
  </operation>
</portType>
<binding name="QuoteServiceSoap" type="s0:QuoteServiceSoap">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document" />
  <operation name="onQuoteReady">
    <soap:operation soapAction="http://workshop.bea.com/QuoteService/onQuoteReady" style="document" />
    <cw:transition phase="finish" />
    <input>
      <soap:body use="literal" />
    </input>
    <output>
      <soap:body use="literal" />
      <soap:header wsdl:required="true" message="s0:CallbackHeader_literal" part="CallbackHeader" />
    </output>
  </operation>
  <operation name="getQuote">
    <soap:operation soapAction="http://workshop.bea.com/QuoteService/getQuote" style="document" />
    <cw:transition phase="start" />
    <input>
      <soap:body use="literal" />
      <soap:header wsdl:required="true" message="s0:StartHeader_literal" part="StartHeader" />
    </input>
    <output>
      <soap:body use="literal" />
    </output>
  </operation>
</binding>
<binding name="QuoteServiceHttpGet" type="s0:QuoteServiceHttpGet">
  <http:binding verb="GET" />
  <operation name="onQuoteReady">
    <http:operation location="/onQuoteReady" />
    <cw:transition phase="finish" />
    <input>
      <mime:mimeXml part="Body" />
    </input>
    <output>
      <http:urlEncoded />
    </output>
  </operation>
  <operation name="getQuote">
    <http:operation location="/getQuote" />
    <cw:transition phase="start" />
    <input>
      <http:urlEncoded />
    </input>
    <output>
      <http:urlEncoded />
    </output>
  </operation>

```

navWebServices.html Sample

```

        </input>
        <output/>
    </operation>
</binding>
<binding name="QuoteServiceHttpPost" type="s0:QuoteServiceHttpPost">
    <http:binding verb="POST"/>
    <operation name="onQuoteReady">
        <http:operation location="/onQuoteReady"/>
        <cw:transition phase="finish"/>
        <input>
            <mime:mimeType part="Body"/>
        </input>
        <output>
            <mime:content type="application/x-www-form-urlencoded"/>
        </output>
    </operation>
    <operation name="getQuote">
        <http:operation location="/getQuote"/>
        <cw:transition phase="start"/>
        <input>
            <mime:content type="application/x-www-form-urlencoded"/>
        </input>
        <output/>
    </operation>
</binding>
<service name="QuoteService">
    <documentation>&lt;p&gt;This is the original service from which QuoteServiceControl.jcx
    <port name="QuoteServiceSoap" binding="s0:QuoteServiceSoap">
        <soap:address location="http://localhost:7001/service/QuoteService.jws"/>
    </port>
    <port name="QuoteServiceHttpGet" binding="s0:QuoteServiceHttpGet">
        <http:address location="http://localhost:7001/service/QuoteService.jws"/>
    </port>
    <port name="QuoteServiceHttpPost" binding="s0:QuoteServiceHttpPost">
        <http:address location="http://localhost:7001/service/QuoteService.jws"/>
    </port>
</service>
</definitions>
* ::
*/

```


timer Samples

This section contains source code for the following samples.

Samples Included in This Section

AdvancedTimer.jws Sample

LegacySystem.jws Sample

LegacySystemControl.jcx Sample

SimpleTimer.jws Sample

AdvancedTimer.jws Sample

This topic includes the source code for the AdvancedTimer.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/timer/

Sample Source Code

```
package timer;

import com.bea.control.JwsContext;
import com.bea.control.TimerControl;
import java.util.Date;

/**
 * <p>A web service that demonstrates an asynchronous interface on a legacy system that
 * does not support asynchrony. It does so by "polling" the legacy system: calling
 * it repeatedly to see if it's done.</p>
 *
 * <p>To use this service, call <b>start</b> to create a new conversational instance and invoke
 * the operation on the legacy system. After the legacy system completes it's
 * operation, we will invoke the client's <b>onDone</b> callback. If the legacy system does
 * not complete in 15 seconds, we will invoke the <b>onDone</b> callback, but with a failure
 * result.</p>
 *
 * @common:target-namespace namespace="http://workshop.bea.com/AdvancedTimer"
 */
public class AdvancedTimer implements com.bea.jws.WebService
{
    /**
     * @common:context
     */
    JwsContext context;

    /**
     * @common:control
     */
    private timer.LegacySystemControl legacySystem;

    /**
     * <p>This is the timer that we use to poll the legacy system. Each time the timer
     * calls us back, we will poll the legacy system again.</p>
     *
     * <p>The timer will expire in 10 seconds the first time, then every second after
     * that.</p>
     *
     * @jc:timer repeats-every="1 seconds" timeout="10 seconds"
     * @common:control
     */
    private TimerControl timer;

    /**
```

navWebServices.html Sample

```
* <p>This variable counts how many times the timer has called us back.  Once it
* has called us 6 times, then 15 seconds have elapsed.</p>
*
* <p>Note that this member variable is made persistent by the presence of
* the <b>@jws:conversation</b> tags.</p>
*/
private int alarmCount;

public Callback callback;

public interface Callback
{
    /**
     * <p>This callback will be invoked when the legacy system completes successfully
     * or if it does not complete in 15 seconds.</p>
     *
     * @param succeeded Indicates whether the legacy system completed its
     *                  operation successfully in time.
     *
     * @jws:conversation phase="finish"
     */
    public void onDone(boolean succeeded);
}

/**
 * <p>Starts a new conversation.  This will result in an <b>onDone</b> callback in
 * 10 to 15 seconds.</p>
 *
 * @throws Exception As thrown by TimerControl.start().
 *
 * @common:operation
 * @jws:conversation phase="start"
 */
public void start() throws Exception
{
    // Start the legacy system working on this operation.
    legacySystem.start();

    // Start the timer going.  It will call us back after 10 seconds, and then
    // once a second after that.
    timer.start();

    // Record that we have not received any onTimeout events yet.
    alarmCount = 0;
}

/**
 * <p>This will be called each time the timer alarm goes off.  Each time we get
 * invoked, we will "poll" the legacy system to see if it has completed.
 * If it has not completed after 15 seconds (6 calls to this method), then
 * we will send back a failure result.</p>
 *
 * @throws Exception This can be thrown by the timer.
 */
private void timer_onTimeout(long time) throws Exception
{
    // Poll the legacy system to see if it has completed.
    if (legacySystem.isDone())
    {
        // Notify our client that the operation completed successfully.
        callback.onDone(true);
    }
}
```

navWebServices.html Sample

```
// Turn the timer off since we no longer need it.
timer.stop();

// terminate the conversation since we no longer need it.
context.finishConversation();
}
else
{
    // Increment the count of times we have received onTimeout events.
    alarmCount += 1;

    // If we have been called 6 times, then 15 seconds have elapsed.
    // In that case, send the operation has failed.
    if (alarmCount == 6)
    {
        // Notify our client that the operation failed.
        callback.onDone(false);

        // Turn off the timer since we no longer need it.
        timer.stop();
    }
}
}
}
```

LegacySystem.jws Sample

This topic includes the source code for the LegacySystem.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/timer/

Sample Source Code

```
package timer;

import java.util.Date;
import com.bea.control.TimerControl;

/**
 * <p>This web service is used by the AdvancedTimer sample. It a mock
 * version of a legacy system that lacks the ability to call you back when its
 * operation is complete. To get around this limitation, AdvancedTimer has to
 * "poll" this service to determine when it's complete.</p>
 *
 * <p>This service has only a single operation that it can perform. To start this
 * operation running, call <b>start</b>. Once <b>start</b> has been called, you can call
 * <b>isDone</b> to determine if the operation has completed.</p>
 *
 * <p>We implement this operation by using a timer. The timer will fire between
 * 10 and 20 seconds after the operation is started. Once the timer goes off,
 * we will report the operation as completed.</p>
 *
 * @common:target-namespace namespace="http://workshop.bea.com/LegacySystem"
 */
public class LegacySystem implements com.bea.jws.WebService
{
    /**
     * <p>This variable stores whether the operation has completed yet.
     * It is made persistent by the presence of a "<b>@common:conversation start</b>" tag.</p>
     */
    private boolean complete;

    /**
     * @jc:timer
     * @common:control
     */
    private com.bea.control.TimerControl timer;

    /**
     * <p>Starts a new instance of the operation running. This operation will
     * complete in between 10 and 15 seconds.</p>
     *
     * @throws Exception As thrown by TimerControl.start or TimerControl.setTimeoutIn.
     *
     * @common:operation
     * @jws:conversation phase="start"
     */
    public void start() throws Exception
```

navWebServices.html Sample

```
{
    int delay = 10 + (int)(11.0 * Math.random());

    // This will set the timeout to occur in between 10 and 20 seconds.
    timer.setTimeout((long)delay);

    // This will tell the timer to call us back in the amount of time that
    // we set above. Don't forget this part!
    timer.start();

    // We will not complete until the timer calls us back.
    complete = false;
}

/**
 * <p>Call this to determine if the operation has completed.</p>
 *
 * @returns Whether the operation has completed when called.
 *
 * @common:operation
 * @jws:conversation phase="continue"
 */
public boolean isDone()
{
    return complete;
}

/** <p>When the timer calls us back, we have completed the operation.</p> */
private void timer_onTimeout(long time)
{
    complete = true;
}

/**
 * @common:operation
 * @jws:conversation phase="finish"
 */
public void finish()
{
}
}
```

LegacySystemControl.jcx Sample

This topic includes the source code for the LegacySystemControl.jcx Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/timer/

Sample Source Code

```
package timer;

/**
 * <p>This web service is used by the AdvancedTimer sample. It a mock version of a legacy system
 * @jc:location http-url="LegacySystem.jws" jms-url="LegacySystem.jws"
 * @jc:wSDL file="#LegacySystemWSDL"
 * @editor-info:link autogen-style="java" source="LegacySystem.jws" autogen="true"
 */
public interface LegacySystemControl extends com.bea.control.ControlExtension, com.bea.control.Control {
    public static class StartHeader
        implements java.io.Serializable
    {
        public java.lang.String conversationID;
        public java.lang.String callbackLocation;
    }

    public static class ContinueHeader
        implements java.io.Serializable
    {
        public java.lang.String conversationID;
    }

    public static class CallbackHeader
        implements java.io.Serializable
    {
        public java.lang.String conversationID;
    }

    /**
     * <p>Starts a new instance of the operation running. This operation will complete in between
     * @jc:conversation phase="start"
     */
    public void start ();

    /**
     * <p>Call this to determine if the operation has completed.</p>
     * @jc:conversation phase="continue"
     */
    public boolean isDone ();

    /**
     * @jc:conversation phase="finish"
     */
}
```

navWebServices.html Sample

```

public void finish ();

static final long serialVersionUID = 1L;
}

/** @common:define name="LegacySystemWsd1" value::
<?xml version="1.0" encoding="utf-8"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:conv="http://www.openuri.org/20
    <types>
        <s:schema elementFormDefault="qualified" targetNamespace="http://workshop.bea.com/LegacySystemWsd1">
            <s:element name="start">
                <s:complexType>
                    <s:sequence/>
                </s:complexType>
            </s:element>
            <s:element name="startResponse">
                <s:complexType>
                    <s:sequence/>
                </s:complexType>
            </s:element>
            <s:element name="isDone">
                <s:complexType>
                    <s:sequence/>
                </s:complexType>
            </s:element>
            <s:element name="isDoneResponse">
                <s:complexType>
                    <s:sequence>
                        <s:element name="isDoneResult" type="s:boolean"/>
                    </s:sequence>
                </s:complexType>
            </s:element>
            <s:element name="boolean" type="s:boolean"/>
            <s:element name="finish">
                <s:complexType>
                    <s:sequence/>
                </s:complexType>
            </s:element>
            <s:element name="finishResponse">
                <s:complexType>
                    <s:sequence/>
                </s:complexType>
            </s:element>
        </s:schema>

        <s:schema elementFormDefault="qualified" targetNamespace="http://www.openuri.org/2002/01/26/conv">
            <s:element name="StartHeader" type="conv:StartHeader"/>
            <s:element name="ContinueHeader" type="conv:ContinueHeader"/>
            <s:element name="CallbackHeader" type="conv:CallbackHeader"/>
            <s:complexType name="StartHeader">
                <s:sequence>
                    <s:element minOccurs="0" maxOccurs="1" name="conversationID" type="s:string"/>
                    <s:element minOccurs="0" maxOccurs="1" name="callbackLocation" type="s:string"/>
                </s:sequence>
            </s:complexType>
            <s:complexType name="ContinueHeader">
                <s:sequence>
                    <s:element minOccurs="1" maxOccurs="1" name="conversationID" type="s:string"/>
                </s:sequence>
            </s:complexType>
            <s:complexType name="CallbackHeader">

```


navWebServices.html Sample

```

    <s:sequence>
      <s:element minOccurs="1" maxOccurs="1" name="conversationID" type="s:string"/>
    </s:sequence>
  </s:complexType>
</s:schema>
</types>
<message name="startSoapIn">
  <part name="parameters" element="s0:start"/>
</message>
<message name="startSoapOut">
  <part name="parameters" element="s0:startResponse"/>
</message>
<message name="isDoneSoapIn">
  <part name="parameters" element="s0:isDone"/>
</message>
<message name="isDoneSoapOut">
  <part name="parameters" element="s0:isDoneResponse"/>
</message>
<message name="finishSoapIn">
  <part name="parameters" element="s0:finish"/>
</message>
<message name="finishSoapOut">
  <part name="parameters" element="s0:finishResponse"/>
</message>
<message name="startHttpGetIn"/>
<message name="startHttpGetOut"/>
<message name="isDoneHttpGetIn"/>
<message name="isDoneHttpGetOut">
  <part name="Body" element="s0:boolean"/>
</message>
<message name="finishHttpGetIn"/>
<message name="finishHttpGetOut"/>
<message name="startHttpPostIn"/>
<message name="startHttpPostOut"/>
<message name="isDoneHttpPostIn"/>
<message name="isDoneHttpPostOut">
  <part name="Body" element="s0:boolean"/>
</message>
<message name="finishHttpPostIn"/>
<message name="finishHttpPostOut"/>
<message name="StartHeader_literal">
  <part name="StartHeader" element="conv:StartHeader"/>
</message>
<message name="ContinueHeader_literal">
  <part name="ContinueHeader" element="conv:ContinueHeader"/>
</message>
<portType name="LegacySystemSoap">
  <operation name="start">
    <documentation><p>Starts a new instance of the operation running. This operation</p></documentation>
    <input message="s0:startSoapIn"/>
    <output message="s0:startSoapOut"/>
  </operation>
  <operation name="isDone">
    <documentation><p>Call this to determine if the operation has completed.</p></documentation>
    <input message="s0:isDoneSoapIn"/>
    <output message="s0:isDoneSoapOut"/>
  </operation>
  <operation name="finish">
    <input message="s0:finishSoapIn"/>
    <output message="s0:finishSoapOut"/>
  </operation>

```

navWebServices.html Sample

```

</portType>
<portType name="LegacySystemHttpGet">
  <operation name="start">
    <documentation><p></p></documentation>
    <input message="s0:startHttpGetIn"/>
    <output message="s0:startHttpGetOut"/>
  </operation>
  <operation name="isDone">
    <documentation><p></p></documentation>
    <input message="s0:isDoneHttpGetIn"/>
    <output message="s0:isDoneHttpGetOut"/>
  </operation>
  <operation name="finish">
    <input message="s0:finishHttpGetIn"/>
    <output message="s0:finishHttpGetOut"/>
  </operation>
</portType>
<portType name="LegacySystemHttpPost">
  <operation name="start">
    <documentation><p></p></documentation>
    <input message="s0:startHttpPostIn"/>
    <output message="s0:startHttpPostOut"/>
  </operation>
  <operation name="isDone">
    <documentation><p></p></documentation>
    <input message="s0:isDoneHttpPostIn"/>
    <output message="s0:isDoneHttpPostOut"/>
  </operation>
  <operation name="finish">
    <input message="s0:finishHttpPostIn"/>
    <output message="s0:finishHttpPostOut"/>
  </operation>
</portType>
<binding name="LegacySystemSoap" type="s0:LegacySystemSoap">
  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
  <operation name="start">
    <soap:operation soapAction="http://workshop.bea.com/LegacySystem/start" style="document"/>
    <cw:transition phase="start"/>
    <input>
      <soap:body use="literal"/>
      <soap:header wsdl:required="true" message="s0:StartHeader_literal" part="StartHeader"/>
    </input>
    <output>
      <soap:body use="literal"/>
    </output>
  </operation>
  <operation name="isDone">
    <soap:operation soapAction="http://workshop.bea.com/LegacySystem/isDone" style="document"/>
    <cw:transition phase="continue"/>
    <input>
      <soap:body use="literal"/>
      <soap:header wsdl:required="true" message="s0:ContinueHeader_literal" part="ContinueHeader"/>
    </input>
    <output>
      <soap:body use="literal"/>
    </output>
  </operation>
  <operation name="finish">
    <soap:operation soapAction="http://workshop.bea.com/LegacySystem/finish" style="document"/>
    <cw:transition phase="finish"/>
    <input>

```

navWebServices.html Sample

```
<soap:body use="literal"/>
<soap:header wsdl:required="true" message="s0:ContinueHeader_literal" part="ContinueHeader"/>
</input>
<output>
  <soap:body use="literal"/>
</output>
</operation>
</binding>
<binding name="LegacySystemHttpGet" type="s0:LegacySystemHttpGet">
  <http:binding verb="GET"/>
  <operation name="start">
    <http:operation location="/start"/>
    <cw:transition phase="start"/>
    <input>
      <http:urlEncoded/>
    </input>
    <output/>
  </operation>
  <operation name="isDone">
    <http:operation location="/isDone"/>
    <cw:transition phase="continue"/>
    <input>
      <http:urlEncoded/>
    </input>
    <output>
      <mime:mimeType part="Body"/>
    </output>
  </operation>
  <operation name="finish">
    <http:operation location="/finish"/>
    <cw:transition phase="finish"/>
    <input>
      <http:urlEncoded/>
    </input>
    <output/>
  </operation>
</binding>
<binding name="LegacySystemHttpPost" type="s0:LegacySystemHttpPost">
  <http:binding verb="POST"/>
  <operation name="start">
    <http:operation location="/start"/>
    <cw:transition phase="start"/>
    <input>
      <mime:content type="application/x-www-form-urlencoded"/>
    </input>
    <output/>
  </operation>
  <operation name="isDone">
    <http:operation location="/isDone"/>
    <cw:transition phase="continue"/>
    <input>
      <mime:content type="application/x-www-form-urlencoded"/>
    </input>
    <output>
      <mime:mimeType part="Body"/>
    </output>
  </operation>
  <operation name="finish">
    <http:operation location="/finish"/>
    <cw:transition phase="finish"/>
    <input>
```

navWebServices.html Sample

```
<mime:content type="application/x-www-form-urlencoded"/>
</input>
<output/>
</operation>
</binding>
<service name="LegacySystem">
  <documentation>&lt;p&gt;This web service is used by the AdvancedTimer sample. It a mock
  <port name="LegacySystemSoap" binding="s0:LegacySystemSoap">
    <soap:address location="http://localhost:7001/timer/LegacySystem.jws"/>
  </port>
  <port name="LegacySystemHttpGet" binding="s0:LegacySystemHttpGet">
    <http:address location="http://localhost:7001/timer/LegacySystem.jws"/>
  </port>
  <port name="LegacySystemHttpPost" binding="s0:LegacySystemHttpPost">
    <http:address location="http://localhost:7001/timer/LegacySystem.jws"/>
  </port>
</service>
</definitions>
* ::
*/
```

SimpleTimer.jws Sample

This topic includes the source code for the SimpleTimer.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/timer/

Sample Source Code

```
package timer;

import com.bea.control.TimerControl;
import java.text.SimpleDateFormat;
import java.util.Date;

/**
 * <p>This web service demonstrates the use of a Timer control.</p>
 *
 * <p>Each time setAlarm is called it sets a timer to expire in 5 seconds. When the
 * timer expires, the onAlarm callback is invoked on the client.</p>
 *
 * <p>Note that the setAlarm method calls restart on the timer, instead of start. This
 * will cause any pending timeout events to be discarded. So if you send three setAlarm
 * messages quickly in sequence, only one onAlarm will occur. This will come five
 * seconds after the last time setAlarm was called.</p>
 *
 * <p>To test this service, invoke the createTimer method in the Test View.
 * This will create a new conversational instance of this class. Next click on the
 * conversation that was created and invoke the setAlarm method. After five seconds
 * have elapsed, right click anywhere in the Test View and select "Refresh".
 * You should see an "onAlarm" callback invocation listed in the test View log.
 * Click on the callback log entry to see the string sent back with the callback.</p>
 *
 * <p>Try sending a few setAlarm messages in sequence. See how many onAlarm
 * callbacks messages are sent.</p>
 *
 * @common:target-namespace namespace="http://workshop.bea.com/SimpleTimer"
 */
public class SimpleTimer implements com.bea.jws.WebService
{
    /**
     * <p>A timer control that will fire an onTimeout event exactly one time 5 seconds
     * after start or restart is called.</p>
     * @jc:timer timeout="5 seconds"
     * @common:control
     */
    private com.bea.control.TimerControl timer;

    private Callback callback;

    public interface Callback
    {
        /**
```

navWebServices.html Sample

```
        * @jws:conversation phase="continue"
        */
    public void onAlarm(String timeFired);
}

/**
 * <p>Creates a new conversational instance of this service.  Once one has been
 * created, you can send setAlarm messages.</p>
 *
 * <p>Click on the Conversation link that comes back from this service to access
 * the continue and finish methods.</p>
 *
 * @common:operation
 * @jws:conversation phase="start"
 */
public void createTimer()
{
    /*
     * Nothing to do.  A new instance is created automatically because of the
     * "@jws:conversation start" tag.  This method is just here to start the
     * conversation.
     */
}

/**
 * <p>Resets the timer control, or starts it the first time.</p>
 *
 * <p>Any pending alarms are discarded, leaving one that will expire according to
 * the @jc:timer tag, above.</p>
 *
 * @common:operation
 * @jws:conversation phase="continue"
 */
public void setAlarm() throws Exception
{
    timer.restart();
}

/**
 * <p>This method will be called by the timer 5 seconds after we start or
 * restart the timer.</p>
 *
 * <p>Invokes the onAlarm callback on the appropriate client.</p>
 *
 * @param time  The time in milliseconds that the timer was scheduled to expire.
 */
private void timer_onTimeout(long time)
{
    /*
     * Use the time received from the callback to create a java.util.Date object.</p>
     *
     * Use a java.text.SimpleDateFormat date formatter to turn the java.util.Date
     * object into a human readable date string.
     */
    SimpleDateFormat formatter = new SimpleDateFormat("yyyy.MM.dd 'at' hh:mm:ss a zzz");
    callback.onAlarm("timer fired on '" + formatter.format(new Date(time)) + "'");
}

/**
 * <p>Destroys this instance of the service.</p>
 */
}
```

navWebServices.html Sample

```
* <p>The <tt>@jws:conversation phase="finish"</tt> annotation on this method
* means that when the method is invoked the conversation it belongs to will
* be terminated.</p>
*
* @common:operation
* @jws:conversation phase="finish"
*/
public void destroyTimer()
{
    /*
    * Nothing to do. This method is just here to terminate the
    * conversation.
    *
    * You should always provide a way for clients to terminate conversations
    * when they are finished. Conversations are resources; if you leave a lot
    * of dangling conversations performance of your server may suffer.
    *
    * The lifetime of a conversation may be set with the
    * @jws:conversation-lifetime tag. Any conversation will terminate
    * automatically if after that time if no conversational activity occurs.
    *
    * Default value is one day.
    */
}
```

xmlBeans Samples

This section contains source code for the following samples.

Samples Included in This Section

cursor Samples

schema Samples

xquery Samples

cursor Samples

This section contains source code for the following samples.

Samples Included in This Section

BatchOrderSimple.xml Sample

InventoryItem.xml Sample

MixedContent.jws Sample

TokenTypes.jws Sample

BatchOrderSimple.xml Sample

This topic includes the source code for the BatchOrderSimple.xml Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/xmlBeans/cursor/

Sample Source Code

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Order summary from the Orders database.-->
<?xml-stylesheet type="text/xsl" href="order_summary.xslt"?>
<batchWidgetOrder batchID="3DJ" xmlns:ns="http://openuri.org/shipping/">
  <order id="13">
    <storeID>ABCD</storeID>
    <shipTo>
      <name>Gladys Kravitz</name>
      <address>1 A St.</address>
      <city>Seattle</city>
      <state>WA</state>
      <zip>98104</zip>
    </shipTo>
  </order>
</batchWidgetOrder>
```

InventoryItem.xml Sample

This topic includes the source code for the InventoryItem.xml Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/xmlBeans/cursor/

Sample Source Code

```
<inventory xmlns="http://openuri.org/bean/samples/workshop/xmlBeans/mixedContent">
  <item id="897946">
    <name>locking flange harbinger</name>
    <description>Completely myopic gyrating mill-flange.</description>
    <price>21.79</price>
    <quantity>4594</quantity>
  </item>
  <item id="745621">
    <name>protaic calliphange</name>
    <description>Asymmetrical flared-gill spongimass. Complements the locking flange</description>
    <price>19.95</price>
    <quantity>2</quantity>
  </item>
  <item id="784269">
    <name>gyrating mill-flange</name>
    <description>Polymorphic atrophical mylobium. Not compatible with any other dev</description>
    <price>.99</price>
    <quantity>1205987</quantity>
  </item>
</inventory>
```

MixedContent.jws Sample

This topic includes the source code for the MixedContent.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/xmlBeans/cursor/

Sample Source Code

```
package xmlBeans.cursor;

import com.bea.xml.XmlCursor;
import org.openuri.bea.samples.workshop.xmlBeans.mixedContent.DescriptionType;
import org.openuri.bea.samples.workshop.xmlBeans.mixedContent.InventoryDocument;
import org.openuri.bea.samples.workshop.xmlBeans.mixedContent.InventoryDocument.Inventory;
import org.openuri.bea.samples.workshop.xmlBeans.mixedContent.ItemType;

/**
 * A web service illustrating how you can use an XML cursor
 * to manipulate the content of an element. While working with
 * strongly-typed XML (in which you are accessing the XML
 * through an API generated from schema) provides easy
 * access for getting and setting the entire value of an
 * element or attribute, it does not easily provide finer
 * grained access to an element's content. This web service
 * shows how you can use an XML cursor to "dive into" an
 * element's content, manipulating it on a character-by-
 * character level. <br/><br/>
 *
 * The code in this web service is designed to look at the
 * description of each item in an inventory list, creating
 * a link wherever the description contains a reference
 * to another item in the inventory list. This alters the
 * description element so that it contains a mix of text and
 * link elements. Such an element is said to have "mixed
 * content."
 *
 * @common:target-namespace namespace="http://workshop.bea.com/MixedContent"
 */
public class MixedContent implements com.bea.jws.WebService
{
    /**
     * Creates links between items in an inventory list by inserting
     * a <link> element for each linked item. An XmlCursor
     * instance passes through each <description> element, looking
     * for text matching the name of an item. <br/><br/>
     *
     * To test this method, paste the contents of the
     * InventoryItem.xml file over the <inventory>
     * element (and its children) provided in by Test View.
     * Click linkItems to view the XML produced by this code.
     *
     * @common:operation
     */
}
```

navWebServices.html Sample

```
public InventoryDocument linkItems(InventoryDocument inventoryDoc)
{
    /**
     * Retrieve the inventory element and get an array of
     * the item elements it contains.
     */
    Inventory inventory = inventoryDoc.getInventory();
    ItemType[] items = inventory.getItemArray();

    /**
     * Loop through the item elements, examining the
     * description for each to see if another inventory item
     * is mentioned.
     */
    for (int i = 0; i < items.length; i++)
    {
        /**
         * Get details about the current item, including
         * its length. This will be used to measure text
         * while exploring the description.
         */
        String itemName = items[i].getName();
        String itemId = new Integer(items[i].getId()).toString();
        int itemCharCount = itemName.length();

        /**
         * Loop through the item descriptions, looking at each
         * for the name of the current item.
         */
        for (int j = 0; j < items.length; j++)
        {
            DescriptionType description = items[j].getDescription();

            /**
             * Insert an XmlCursor instance and set it at
             * the beginning of the description element's text,
             * just after the start tag.
             */
            XmlCursor cursor = description.newCursor();
            cursor.toLastAttribute();
            cursor.nextToken();

            /**
             * Get a String containing the characters to the
             * immediate right of the cursor, up to the next
             * token (in this case, the next element after
             * the description element). Get the number of
             * characters to the right of the cursor; this will
             * be used to mark the distance the cursor should move
             * before trying another item's description. Also,
             * create a charCount variable to mark the cursor's
             * current position.
             */
            String cursorChars = cursor.getChars();
            int descCharCount = cursorChars.length();
            int charCount = 0;

            /**
             * As long as the cursor hasn't reached the end of the
             * description text, check to see if the text to the
             * cursor's immediate right matches the item name sought.
            */
        }
    }
}
```

navWebServices.html Sample

```
* If it does match, remove the text and create a link
* element to replace it.
*/
while (charCount < descCharCount)
{
    /**
     * A char array to hold the characters currently being
     * checked.
     */
    char[] chars = new char[itemCharCount];

    /**
     * Pass the char array with the getChars method. This
     * method will find the chars from the cursor's
     * immediate right to the char at itemCharCount (the
     * length of the item name currently sought). The
     * method's second argument indicates where in the char
     * array the found text should begin -- here, at the
     * beginning.
     */
    int charsReturned = cursor.getChars(chars, 0, itemCharCount);

    /**
     * If the characters in chars match the item name, then
     * make a link from the text.
     */
    if (new String(chars).equals(itemName))
    {
        /**
         * Remove the found item name.
         */
        cursor.removeChars(itemCharCount);

        /**
         * Begin a new link element whose namespace is the
         * same as the rest of the inventory document. The
         * beginElement method essentially creates a new
         * element with the name specified around the current
         * cursor.
         */
        cursor.beginElement("link",
            "http://openuri.org/bea/samples/workshop/xmlBeans/mixedContent");

        /**
         * Insert an id attribute and make its value the id of
         * the item sought.
         */
        cursor.insertAttributeWithValue("id", itemId);

        /**
         * Insert the item name as the element's value.
         */
        cursor.insertChars(itemName);
    }

    /**
     * Move on to the next character in the description.
     */
    cursor.moveToNextChar(1);

    /**
     * Increment the counter tracking the cursor's position.
     */
    charCount++;
}
/**
```

navWebServices.html Sample

```
        * Be sure to dispose of a cursor that's no longer needed.
        * This allows it to be garbage collected.
        */
        cursor.dispose();
    }
}
/**
 * Return the edited document.
 */
return inventoryDoc;
}
}
```

TokenTypes.jws Sample

This topic includes the source code for the TokenTypes.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/xmlBeans/cursor/

Sample Source Code

```
package xmlBeans.cursor;

import com.bea.xml.XmlCursor;
import com.bea.xml.XmlCursor.TokenType;
import com.bea.xml.XmlObject;
import com.bea.xml.XmlOptions;

/**
 * This web service illustrates how tokens correspond to portions
 * of an XML document. A token represents a logical piece of XML, such
 * as the start or end of an element, text, a comment, and so on. A token
 * type, on the other hand, represents a kind of token. In other words, a token
 * is a STARTDOC (the very beginning of an XML instance, before all markup),
 * a START (the start of an element), a COMMENT, and so on.
 *
 * When navigating XML with a cursor, you move the cursor move past tokens using
 * methods such as toNextToken, toNextElement, and so on. It's important to remember
 * that a cursor is almost always immediately before some token. When at a START token,
 * for example, the cursor is just before the start of an element. This also means
 * that it is before the start of an element and after some other token. In other
 * words, calling currentTokenType on an XmlCursor instance will usually return the
 * token that it is immediately before. The exception to this rule is for an
 * ENDDOC token, which is after all other markup. Because it is at the very
 * end, it can't be before anything else.
 *
 * @common:target-namespace namespace="http://workshop.bea.com/TokenTypes"
 */
public class TokenTypes implements com.bea.jws.WebService
{
    /**
     * This method accepts a well-formed XML document and loops through it to
     * discover the token types and the XML corresponding to each token. Because whitespace
     * is represented by the TEXT token type, some of the tokens found in the
     * document may appear to represent nothing at all. Whitespace can occur
     * between elements.<br/><br/>
     *
     * This method calls a private function that does the actual work to
     * identify the tokens.<br/><br/>
     *
     * To test this method, paste the contents of the BatchOrderSimple.xml file
     * in place of <AnyElement/> in the box provided on the Test XML of Test View.
     *
     * @common:operation
     */
}
```


navWebServices.html Sample

```
public String discoverTokenTypes(XmlObject xmlDoc) throws Exception
{
    XmlCursor documentCursor = xmlDoc.newCursor();
    StringBuffer responseBuffer = new StringBuffer();

    /*
     * Loop through the document, passing the cursor when it stops at each token
     * to the function designed to discover the token type.
     */
    while (documentCursor.hasNextToken()) {
        String tokenInfo = collectTokenTypeInfo(documentCursor) + "\n\n";
        responseBuffer.append(tokenInfo);
        documentCursor.moveToNextToken();
    }
    documentCursor.dispose();
    return responseBuffer.toString();
}

/*
 * Use the int value associated with each token type in a switch statement.
 * The result is to look at each token, discover its type, and return the
 * correct response for addition to a string buffer.
 */
private String collectTokenTypeInfo(XmlCursor cursor){
    String response = null;
    XmlOptions options = new XmlOptions();

    switch (cursor.currentTokenType().intValue()){

        case TokenType.INT_STARTDOC:
            response = cursor.currentTokenType() +
                "; cursor is at the very beginning of the document, before any markup.";
            break;

        case TokenType.INT_START:
            response = cursor.currentTokenType() +
                "; cursor is just before this element's start: \n" + cursor.xmlText(options);
            break;

        case TokenType.INT_ATTR:
            response = cursor.currentTokenType() +
                "; cursor is just before this attribute: \n" + cursor.getTextValue();
            break;

        case TokenType.INT_TEXT:
            response = cursor.currentTokenType() +
                "; cursor is just before this text (may be whitespace): \n" + cursor.getChar();
            break;

        case TokenType.INT_NAMESPACE:
            response = cursor.currentTokenType() +
                "; cursor is just before this namespace: \n" + cursor.xmlText();
            break;

        case TokenType.INT_COMMENT:
            response = cursor.currentTokenType() +
                "; cursor is just before this comment: " + cursor.xmlText();
            break;

        case TokenType.INT_PROCINST:
            response = cursor.currentTokenType() +
```

navWebServices.html Sample

```
        "; cursor is just before this processing instruction: " + cursor.xmlText();
    break;

    case TokenType.INT_END:
        response = cursor.currentTokenType() +
            "; cursor is just before the token representing an element's end.";
        break;

    case TokenType.INT_ENDDOC:
        response = cursor.currentTokenType() +
            "; cursor is at the end of the XML.";
        break;

    case TokenType.INT_NONE:
        response = cursor.currentTokenType() +
            "; cursor is not at any recognizable token; it's just before: \n " +
            cursor.xmlText(options);
        break;
    default:
        response = "Yo! Something funky happened!";
}

/* Return the response string. */
return response;
}
}
```

schema Samples

This section contains source code for the following samples.

Samples Included in This Section

PriceSummary.xml Sample

PurchaseOrder.xml Sample

PurchaseOrderSimple.xml Sample

SchemaChoice.jws Sample

SchemaEnum.jws Sample

SimpleAccess.jws Sample

ThresholdEnums.java Sample

ThresholdEnumsImpl.jcs Sample

ThresholdService.jws Sample

TypeHierarchyPrinter.java Sample

TypeHierarchyPrinterService.jws Sample

XsdConfig.jws Sample

PriceSummary.xml Sample

This topic includes the source code for the PriceSummary.xml Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/xmlBeans/schema/

Sample Source Code

```
<price-summary xmlns="http://openuri.org/easypo/price-summary">
  <price threshold="Above20Dollars">
    <item>
      <title>Burnham's Celestial Handbook, Vol 1</title>
      <amount>21.79</amount>
      <quantity>1</quantity>
    </item>
  </price>
  <price threshold="Between10And20Dollars">
    <item>
      <title>Burnham's Celestial Handbook, Vol 2</title>
      <amount>19.89</amount>
      <quantity>2</quantity>
    </item>
    <item>
      <title>Burnham's Celestial Handbook, Vol 3</title>
      <amount>19.89</amount>
      <quantity>1</quantity>
    </item>
  </price>
</price-summary>
```

PurchaseOrder.xml Sample

This topic includes the source code for the PurchaseOrder.xml Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/xmlBeans/schema/

Sample Source Code

```
<purchase-order xmlns="http://openuri.org/easypo">
  <customer>
    <name>Gladys Kravitz</name>
    <address>Anytown, PA</address>
  </customer>
  <date>2001-12-17T09:30:47-05:00</date>
  <line-item>
    <description>Burnham's Celestial Handbook, Vol 1</description>
    <per-unit-ounces>5</per-unit-ounces>
    <price>21.79</price>
    <quantity>2</quantity>
  </line-item>
  <line-item>
    <description>Burnham's Celestial Handbook, Vol 2</description>
    <per-unit-ounces>5</per-unit-ounces>
    <price>19.89</price>
    <quantity>2</quantity>
  </line-item>
  <line-item>
    <description>Burnham's Celestial Handbook, Vol 3</description>
    <per-unit-ounces>5</per-unit-ounces>
    <price>19.89</price>
    <quantity>1</quantity>
  </line-item>
  <shipper>
    <name>UPS</name>
    <per-ounce-rate>0.74</per-ounce-rate>
  </shipper>
</purchase-order>
```

PurchaseOrderSimple.xml Sample

This topic includes the source code for the PurchaseOrderSimple.xml Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/xmlBeans/schema/

Sample Source Code

```
<purchase-order xmlns="http://openuri.org/easypo">
  <customer>
    <name>Gladys Kravitz</name>
    <address>Anytown, PA</address>
  </customer>
  <date>2001-12-17T09:30:47-05:00</date>
  <line-item>
    <description>Burnham's Celestial Handbook, Vol 1</description>
    <per-unit-ounces>5</per-unit-ounces>
    <price>21.79</price>
    <quantity>2</quantity>
  </line-item>
  <shipper>
    <name>UPS</name>
    <per-ounce-rate>0.74</per-ounce-rate>
  </shipper>
</purchase-order>
```

SchemaChoice.jws Sample

This topic includes the source code for the SchemaChoice.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/xmlBeans/schema/

Sample Source Code

```
package xmlBeans.schema;

import org.openuri.bea.samples.workshop.xmlBeans.maritalStatus.EmployeeDocument.Employee;
import org.openuri.bea.samples.workshop.xmlBeans.maritalStatus.EmployeeDocument;
import org.openuri.bea.samples.workshop.xmlBeans.maritalStatus.MaritalStatusDocument.MaritalSta
import org.openuri.bea.samples.workshop.xmlBeans.maritalStatus.MaritalStatusDocument;
import com.bea.xml.XmlOptions;
import com.bea.xml.XmlError;
import java.io.File;
import java.util.ArrayList;

/**
 * This web services illustrates how you can construct an XML document
 * with XMLBeans and types compiled from schema. It also shows how you
 * can validate the document in cases where your code may create a
 * document that is invalid. <br/><br/>
 *
 * The schema for the document created here, EmployeeMarital.xsd,
 * defines a snippet of data about an employee that specifies their
 * marital status (eg, "Married", "Single", or "Domestic partner").
 * The schema defines a "choice" structure in which only one of a
 * set of elements is allowed. The set in this case includes
 * partner-name and spouse-name elements. Only one of these would
 * be applicable for a given employee, so only one should be allowed.
 * However, to illustrate validation, the code here tries to add both. <br/><br/>
 *
 * Validation is done here with the XmlObject.validate method, which in
 * this case takes an XmlOptions argument. The option specifies an ArrayList
 * as a "listener" to use for collecting errors that occur while validating --
 * but any object implementing the java.util.Collection interface will work
 * for this purpose. During validation, the XMLBeans runtime components call
 * the listener's add method as errors are discovered, adding XmlError instances
 * that this code can use to record and report what happened with the validation.
 *
 * @common:target-namespace namespace="http://workshop.bea.com/SchemaChoice"
 */
public class SchemaChoice implements com.bea.jws.WebService
{
    /**
     * This method creates a new employee XML document, inserting first and
     * last name, marital status, and the name of a partner, if any; it also
     * writes an error message to the WebLogic Server console window.
     * The error demonstrates how your code can listen for errors while
     * validating XML.<br/><br/>

```

navWebServices.html Sample

```
*
* To test this method, start enter values into the four fields
* provided. The "maritalStatus" field must be "Married", "Single",
* or "Domestic partner". To "fix" this code, comment the call to
* the setPartnerName OR setSpouseName method.
*
* @common:operation
*/
public EmployeeDocument updateEmployData(String firstName, String lastName,
    String maritalStatus, String otherName) throws Exception
{
    /**
     * Create a new employee XML document.
     */
    EmployeeDocument empDoc = EmployeeDocument.Factory.newInstance();
    Employee newEmp = empDoc.addNewEmployee();

    // Set the first and last names.
    newEmp.setFirstName(firstName);
    newEmp.setLastName(lastName);

    if (maritalStatus.equals("Married"))
    {
        newEmp.setMaritalStatus(MaritalStatus.MARRIED);
    }
    else if (maritalStatus.equals("Single"))
    {
        newEmp.setMaritalStatus(MaritalStatus.SINGLE);
    }
    else if (maritalStatus.equals("Domestic partner"))
    {
        newEmp.setMaritalStatus(MaritalStatus.DOMESTIC_PARTNER);
    }
    else
    {
        throw new Exception("Exception: Invalid marital status given.");
    }

    /**
     * Create an XmlOptions instance to use with the validate
     * method. Add to the instance an ArrayList to use as an error
     * listener for recording errors as validation occurs. But any
     * object implementing the Collection interface will do here.
     */
    XmlOptions validateOptions = new XmlOptions();
    ArrayList errorList = new ArrayList();
    validateOptions.put(XmlOptions.ERROR_LISTENER, errorList);

    /**
     * INVALID. This code sets both the partner-name and
     * spouse-name elements, which is illegal. According to the
     * schema, only one of these is allowed. Note that the
     * action is allowed in code because it is possible that
     * the XML may go through multiple invalid states before the
     * code is finished assembling it. The validate method is called
     * at the end to ensure that the document is valid.
     */
    newEmp.setPartnerName(otherName);
    newEmp.setSpouseName(otherName);

    /**
```


navWebServices.html Sample

```
* Is the XML valid?
*/
boolean isValid = newEmp.validate(validateOptions);

/**
 * If the new XML is not valid, output the errors to
 * the WLS console. Of course, you could also print the
 * messages to a log. While this code returns the newly
 * constructed XML whether valid or not, it might instead
 * return an error if the XML was invalid.
 */
if (!isValid)
{
    for (int i = 0; i < errorList.size(); i++)
    {
        /**
         * Retrieve XmlError instances from the ArrayList.
         * These contain information about the context of
         * the validation error.
         */
        XmlError error = (XmlError)errorList.get(i);

        /**
         * Print information contained in the XmlError instance.
         */
        System.out.println("\n");
        System.out.println("Message: " + error.getMessage() + "\n");
        System.out.println("Location of invalid XML: " +
            error.getCursorLocation().xmlText() + "\n");
    }
}
return empDoc;
}
```

SchemaEnum.jws Sample

This topic includes the source code for the SchemaEnum.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/xmlBeans/schema/

Sample Source Code

```
package xmlBeans.schema;

import org.openuri.easypo.PurchaseOrderDocument;
import org.openuri.easypo.PurchaseOrderDocument.PurchaseOrder;
import org.openuri.easypo.priceSummary.PriceSummaryDocument;
import org.openuri.easypo.priceSummary.PriceSummaryDocument.PriceSummary;
import org.openuri.easypo.priceSummary.PriceType;
import org.openuri.easypo.priceSummary.ItemType;
import org.openuri.easypo.LineItem;

/**
 * This web service illustrates how you can access XML values that are
 * defined in schema as enumerations. When a schema containing
 * enumerations is compiled, the generated Java types represent the
 * enumerations with enumerations of their own. You can access these through
 * their constants and corresponding int values.
 *
 * The schemas used by this service are defined in PriceSummary.xsd and
 * EasyPO.xsd.
 *
 * @common:target-namespace namespace="http://workshop.bea.com/SchemaEnum"
 */
public class SchemaEnum implements com.bea.jws.WebService
{
    /**
     * This method uses values in the incoming XML to construct
     * a new XML document of a different schema. PriceSummary.xsd, the schema
     * for the new document, defines XML enumerations for a price
     * threshold attribute. Items whose price is between $10 and $20 receive
     * a threshold value of 10.00; items above 20 get a threshold value of
     * 20.00.<br/><br/>
     *
     * This method loops through the purchase order items, creating a summary
     * document that specifies their threshold value.<br/><br/>
     *
     * Test this method by pasting the contents of PurchaseOrder.xml into
     * the following box. Be sure to paste the test XML in place of the
     * XML between the <summarizeItems> element tags. Verify
     * the test by comparing the resulting XML with the XML in PriceSummary.xml.<br/><br/>
     *
     * You can use this test's return value in Test View to test
     * the sortByThreshold method.
     *
     * @common:operation
    
```

navWebServices.html Sample

```
*/
public PriceSummaryDocument summarizeItems(PurchaseOrderDocument poDoc)
{
    PurchaseOrder po = poDoc.getPurchaseOrder();

    /*
     * Create a new instance of the PriceSummary schema. This is the document
     * the code creates, extracting values from the purchase order.
     */
    PriceSummaryDocument summaryDoc = PriceSummaryDocument.Factory.newInstance();
    PriceSummary summary = summaryDoc.addNewPriceSummary();

    /*
     * Create price elements to hold item elements according to their
     * price threshold.
     */
    PriceType priceZero = summary.addNewPrice();
    PriceType priceTen = summary.addNewPrice();
    PriceType priceTwenty = summary.addNewPrice();

    /*
     * Set the threshold attribute value for the two new elements.
     */
    priceZero.setThreshold(PriceType.Threshold.BELOW_10_DOLLARS);
    priceTen.setThreshold(PriceType.Threshold.BETWEEN_10_AND_20_DOLLARS);
    priceTwenty.setThreshold(PriceType.Threshold.ABOVE_20_DOLLARS);

    /*
     * Loop through the purchase order line-item elements. If their
     * price child element is between 10.00 and 20.00, add the line-item
     * to the price element whose threshold is 10.00. For those over 20.00,
     * add them to the price element whose threshold is 20.00.
     */
    for (int i = 0; i < po.getLineItemArray().length; i++)
    {
        LineItem item = po.getLineItemArray(i);

        if (item.getPrice() < 10.00)
        {

            ItemType newItem = priceZero.addNewItem();
            newItem.setTitle(item.getDescription());
            newItem.xsetQuantity(item.xgetQuantity());
            newItem.setAmount(item.getPrice());

        } else if (item.getPrice() >= 10.00 && item.getPrice() < 20.00)
        {

            ItemType newItem = priceTen.addNewItem();
            newItem.setTitle(item.getDescription());
            newItem.xsetQuantity(item.xgetQuantity());
            newItem.setAmount(item.getPrice());

        } else if (item.getPrice() >= 20.00)
        {

            ItemType newItem = priceTwenty.addNewItem();
            newItem.setTitle(item.getDescription());
            newItem.xsetQuantity(item.xgetQuantity());
            newItem.setAmount(item.getPrice());

        }
    }
}
```

navWebServices.html Sample

```

    }

    /* Return the newly created summary document. */
    return summaryDoc;
}

/**
 * The sortByThreshold method loops through a price summary to
 * create a string that lists the items grouped by threshold.
 * Unlike the summarizeItems method, with creates a new XML
 * document that contains an attribute whose value is enumerated,
 * this method retrieves values from an enumeration.<br/><br/>
 *
 * This method illustrates how you can use the int value corresponding
 * to enumerations to specify them in Java switch statements.<br/><br/>
 *
 * To test this method, copy the contents of PriceSummary.xml into the
 * box in Test View. Be sure to paste the test XML in place of the XML
 * between the <sortByThreshold> element tags.
 *
 * @common:operation
 */
public String sortByThreshold(PriceSummaryDocument summaryDoc)
{
    System.out.println(summaryDoc.xmlText());
    /*
     * Extract the summary element from the incoming XML, then use it
     * to extract an array of the price elements.
     */
    PriceSummary summary = summaryDoc.getPriceSummary();
    StringBuffer responseBuffer = new StringBuffer();
    PriceType[] priceElements = summary.getPriceArray();

    /*
     * Create string buffers to hold the sorted results of the values
     * retrieved.
     */
    StringBuffer zeroBuffer = new StringBuffer("\n Items under 10 dollars: \n");
    StringBuffer tenBuffer = new StringBuffer("\n Items between 10 and 20 dollars: \n");
    StringBuffer twentyBuffer = new StringBuffer("\n Items more than 20 dollars: \n");

    /*
     * Loop through the price elements, extracting the array of item child
     * elements in each.
     */
    for (int i = 0; i < priceElements.length; i++)
    {
        ItemType[] itemElements = priceElements[i].getItemArray();

        /*
         * Loop through the item elements, binding a variable to each
         * item element in turn.
         */
        for (int j = 0; j < itemElements.length; j++)
        {
            ItemType item = itemElements[j];

            /*
             * For each item element, find out the int value of its price parent
             * element's threshold attribute value. Append the item's title to
             * the appropriate string buffer.

```

navWebServices.html Sample

```
        */
switch(priceElements[i].getThreshold().intValue())
{
    case PriceType.Threshold.INT_BELOW_10_DOLLARS:
        zeroBuffer.append(item.getTitle() + "\n");
        break;

    case PriceType.Threshold.INT_BETWEEN_10_AND_20_DOLLARS:
        tenBuffer.append(item.getTitle() + "\n");
        break;

    case PriceType.Threshold.INT_ABOVE_20_DOLLARS:
        twentyBuffer.append(item.getTitle() + "\n");
        break;

    default:
        System.out.println("Yo! Something unexpected happened!");
        break;
}
    }
}

/*
 * Combine the two result buffers into one, then return a string from the
 * combined result.
 */
responseBuffer.append(tenBuffer);
responseBuffer.append(twentyBuffer);
return responseBuffer.toString();
}
}
```

SimpleAccess.jws Sample

This topic includes the source code for the SimpleAccess.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/xmlBeans/schema/

Sample Source Code

```
package xmlBeans.schema;

import org.openuri.easypo.PurchaseOrderDocument;
import org.openuri.easypo.PurchaseOrderDocument.PurchaseOrder;
import org.openuri.easypo.LineItem;
import java.math.BigDecimal;

/**
 * This web service illustrates basic use of XMLBeans compiled from
 * XML schema. With XMLBeans, you can compile XML schemas to generate
 * types through which you can access XML instances that conform to
 * the schema. With the generated types, you can use get and set methods
 * in the way that you would with other JavaBeans.
 *
 * In WebLogic Workshop, you can compile schema files simply by copying
 * them to a Schemas project in the application. WebLogic Workshop compiles
 * the schemas and places the resulting JAR file among the application's Libraries.
 * From there, types generated from the schema are available to other code in
 * the application.
 *
 * The schema that supports the XML used by this web service is in EasyPO.xsd.
 * The PurchaseOrderDocument type that is a parameter for each of this
 * web service's methods is a type generated from the schema. It represents
 * the XML document at the top level of the incoming purchase order XML. WebLogic
 * Workshop automatically loads the incoming XML (here, the test XML) into
 * the type specified in the parameter.
 *
 * @common:target-namespace namespace="http://workshop.bea.com/SimpleAccess"
 */
public class SimpleAccess implements com.bea.jws.WebService
{
    /**
     * This method uses simple XMLBean get methods to retrieve values
     * from a purchase order XML document. All of the retrieved values
     * are put into a StringBuffer object, which is converted to a String
     * for display in Test View.<br/><br/>
     *
     * To test this method, copy the contents of the PurchaseOrderSimple.xml
     * file in this project and paste them into the box below. Be sure to
     * paste the test XML in place of the XML between the <getSingleItemDetails>
     * element tags.
     *
     * @common:operation
     */
}
```

navWebServices.html Sample

```

*/
public String getSingleItemDetails(PurchaseOrderDocument poDoc)
{
    // A buffer to hold the response as it's accumulated.
    StringBuffer responseBuffer = new StringBuffer();

    /*
     * Use the top-level document to get the top-level element,
     * purchase-order.
     */
    PurchaseOrder po = poDoc.getPurchaseOrder();

    /*
     * A purchase-order element can have multiple line-item child
     * elements. The PurchaseOrder type that results from
     * compiling the schema reflects this by providing get and set
     * methods for handling its element children as an array.
     */
    LineItem[] lineItems = po.getLineItemArray();

    // Get the first line-item element in order to retrieve its values.
    LineItem lineItem = lineItems[0];

    // Append the value of each line-item element to the string buffer.
    responseBuffer.append("\n");
    responseBuffer.append("Description: " + lineItem.getDescription() + "\n");
    responseBuffer.append("Quantity: " + lineItem.getQuantity() + "\n");
    responseBuffer.append("Price: " + lineItem.getPrice() + "\n");

    // Return the buffer for display in Test View.
    return responseBuffer.toString();
}

/**
 * This method illustrates how you can use XMLBeans to set values
 * in an XML instance document. As with the getSingleItemDetails
 * method, you use JavaBeans accessors provided by the types
 * generated from schema.<br/><br/>
 *
 * To test this method, copy the contents of PurchaseOrderSimple.xml
 * into the box below. Be sure to paste the test XML in place of the
 * XML that begins and ends with <?xml:purchase-order> tags. In
 * other words, replace all of the elements whose namespace prefix is
 * eas. After pasting in the test XML, enter new values for the
 * newDescription, newPerUnitOunce, newQuantity, and newPrice
 * elements at the bottom. Test View has provided these as placeholders.<br/><br/>
 *
 * This method returns the same object that it accepts, but with the
 * new values you specify. Test View displays the returned object
 * as its XML value.
 *
 * @common:operation
 */
public PurchaseOrderDocument setSingleItemDetails(PurchaseOrderDocument poDoc,
    String newDescription, double newPerUnitOunces,
    int newQuantity, double newPrice)
{
    /*
     * Get the purchase-order element contained in the incoming
     * XML document.
     */

```

navWebServices.html Sample

```
PurchaseOrder po = poDoc.getPurchaseOrder();

// Get the line-item element whose values will be changed.
LineItem lineItem = po.getLineItemArray(0);

/*
 * Set new values for each the line-item element's children.
 */
lineItem.setDescription(newDescription);
lineItem.setPerUnitOunces(new BigDecimal(newPerUnitOunces));
lineItem.setQuantity(newQuantity);
lineItem.setPrice(newPrice);

// Return the XML with its new values.
return poDoc;
}

/**
 * This method is very similar to the getSingleItemDetails method,
 * but differs in that it retrieves the values of ALL line-item elements
 * in the incoming XML. It also counts the total quantity for each
 * item order, then calculates a total price.<br/><br/>
 *
 * To test this method, copy the contents of the PurchaseOrderSimple.xml
 * file in this project and paste them into the box below. Be sure to
 * paste the test XML in place of the XML between the <getAllItemDetails>
 * element tags.
 *
 * @common:operation
 */
public String getAllItemDetails(PurchaseOrderDocument poDoc)
{
    StringBuffer responseBuffer = new StringBuffer();

    LineItem[] lineItems = poDoc.getPurchaseOrder().getLineItemArray();
    responseBuffer.append("\n Purchase order has " +
        lineItems.length + " line items. \n");

    // Variables to hold count of item orders and total amount of purchase.
    int numberOfItems = 0;
    double totalAmount = 0.0;

    /*
     * Loop through the list of line-item elements, adding their
     * details to the string buffer.
     */
    for (int j = 0; j < lineItems.length; j++) {
        responseBuffer.append(" Line item: " + j + "\n");
        responseBuffer.append("   Description: " + lineItems[j].getDescription() + "\n");
        responseBuffer.append("   Quantity: " + lineItems[j].getQuantity() + "\n");
        responseBuffer.append("   Price: " + lineItems[j].getPrice() + "\n");

        // Increment item quantity and total purchase amount for each line-item.
        numberOfItems += lineItems[j].getQuantity();
        totalAmount += lineItems[j].getPrice() * lineItems[j].getQuantity();
    }
    responseBuffer.append("Total items: " + numberOfItems + "\n");
    responseBuffer.append("Total amount: " + totalAmount + "\n");

    return responseBuffer.toString();
}
```


}

ThresholdEnums.java Sample

This topic includes the source code for the ThresholdEnums.java Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/xmlBeans/schema/

Sample Source Code

```
package xmlBeans.schema;

import com.bea.control.Control;

/**
 * The Java control interface for the ThresholdEnums
 * control.
 */
public interface ThresholdEnums extends Control
{
    /**
     * Simpler code to provide a list of the enumerations defined in the threshold attribute of
     */
    java.lang.String[] getThresholdValuesSimple();

    /**
     * Provides a list of the enumerations defined in the threshold attribute of the price-summ
     */
    java.lang.String[] getThresholdValues();
}
```

ThresholdEnumsImpl.jcs Sample

This topic includes the source code for the ThresholdEnumsImpl.jcs Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/xmlBeans/schema/

Sample Source Code

```
package xmlBeans.schema;

import com.bea.control.*;
import com.bea.xml.SchemaStringEnumEntry;
import com.bea.xml.SchemaType;
import com.bea.xml.SchemaTypeSystem;
import com.bea.xml.XmlAnySimpleType;
import java.util.HashMap;
import javax.xml.namespace.QName;
import org.openuri.easypo.priceSummary.PriceSummaryDocument;

/**
 * A Java control illustrating how you can use the schema type system to
 * retrieve information about a compiled schema. This service simply
 * retrieves the values of an enumeration defined in schema.
 * <br/><br/>
 * The schema type system API is a powerful way to get information
 * about compiled schema. It is, in a sense, an API about your schema
 * API. The SchemaTypeSystem object represents compiled schemas within reach
 * of your code -- for example, schemas in JAR files in your classpath.
 * <br/><br/>
 * This code is in a Java control, and you can imagine how you might
 * use such a control to provide a list of enumerations for a drop-down
 * list in a JSP page. This control's client, however, is a web service.
 * Test this code by running the ThresholdService.jws.
 *
 * @jcs:jc-jar label="ThresholdEnums"
 * @editor-info:code-gen control-interface="true"
 */
public class ThresholdEnumsImpl implements ThresholdEnums, ControlSource
{
    /**
     * Provides a list of the enumerations defined in the threshold
     * attribute of the price-summary element. The schema in use here
     * is PriceSummary.xsd, in the Schemas project of the SamplesApp
     * application.
     * <br/><br/>
     * This code is overly complex to show some of the things you can
     * do to access a schema type system. There's a simpler version in
     * the getThresholdValuesSimple method.
     *
     * @common:operation
     */
    public String[] getThresholdValues()
    {
```

navWebServices.html Sample

```
// Create a String array to contain the returned values.
String[] enumStrings = new String[0];

/**
 * Create a new instance of PriceSummaryDocument; this will be used to
 * get information about the type system.
 */
PriceSummaryDocument summaryDoc = PriceSummaryDocument.Factory.newInstance();

/**
 * Use the PriceSummaryDocument to get a SchemaTypeSystem object. This
 * object provides access to all compiled schema types available to this
 * code.
 */
SchemaTypeSystem typeSystem = summaryDoc.schemaType().getTypeSystem();

/**
 * Get a SchemaType for the price-summary element. This is necessary because
 * the attribute containing the enumeration isn't available by name because it's
 * anonymous. So it's necessary to go through the type that contains it.
 */
SchemaType type = typeSystem.findType(new QName("http://openuri.org/easypo/price-summary",
    "priceType"));

/**
 * Get all the anonymous types in price-summary, just to iterate through them.
 */
SchemaType[] anonTypes = type.getAnonymousTypes();

/**
 * Loop through the anonymous types, examining their enumerations.
 * Once one is found that belong to a threshold attribute, add the
 * String value for each of its entries to a String array.
 */
for (int i = 0; i < anonTypes.length; i++)
{
    /**
     * Get an array of the enumeration entries in the current type.
     */
    SchemaStringEnumEntry[] enumEntries = anonTypes[i].getStringEnumEntries();

    /**
     * Reassign the response String array with an array of the correct size,
     * now that that size is known.
     */
    enumStrings = new String[enumEntries.length];

    /**
     * Loop through the enumeration entries, adding to the response array those
     * whose container field is "threshold".
     */
    for (int j = 0; j < enumEntries.length; j++)
    {
        if (anonTypes[i].getContainerField().getName().getLocalPart().equals("threshold")
        {
            enumStrings[j] = enumEntries[j].getString();
        }
    }
}
// Return the response array.
return enumStrings;
```

navWebServices.html Sample

```
}

/**
 * Simpler code to provide a list of the enumerations defined in the threshold
 * attribute of the price-summary element. The schema in use here
 * is PriceSummary.xsd, in the Schemas project of the SamplesApp
 * application.
 *
 * @common:operation
 */
public String[] getThresholdValuesSimple()
{
    /**
     * Create a SchemaType object that represents the threshold attribute.
     */
    SchemaType enumType = org.openuri.easypo.priceSummary.PriceType.Threshold.type;

    /**
     * Get the enumeration entries as an array.
     */
    SchemaStringEnumEntry[] enumEntries = enumType.getStringEnumEntries();

    // Create a String array to contain the returned values.
    String[] enumStrings = new String[enumEntries.length];

    /**
     * Loop through the enumeration entries, adding to the response array those
     * whose container field is "threshold".
     */
    for (int j = 0; j < enumEntries.length; j++)
    {
        enumStrings[j] = enumEntries[j].getString();
    }

    // Return the response array.
    return enumStrings;
}
}
```

ThresholdService.jws Sample

This topic includes the source code for the ThresholdService.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/xmlBeans/schema/

Sample Source Code

```
package xmlBeans.schema;

import java.util.HashMap;

/**
 * A web service to use as a test client for the
 * ThresholdEnums Java control. That control illustrates
 * how to use the schema type system to retrieve information
 * about compiled XML schemas.
 *
 * @common:target-namespace namespace="http://workshop.bea.com/ThresholdService"
 */
public class ThresholdService implements com.bea.jws.WebService
{
    /**
     * @common:control
     */
    private xmlBeans.schema.ThresholdEnums thresholdEnums;

    /**
     * Tests the ThresholdEnums control getThresholdValues method.
     * To test this method, simply click the getValues button in
     * Test View. Test View will refresh to display a list of the
     * enumeration values available for the threshold attribute
     * defined in the PriceSummary.xsd schema file. That file is in the
     * Schemas project of the SamplesApp application.
     * <br/><br/>
     * The getValuesSimple method returns the same result, but does
     * so with less code.
     *
     * @common:operation
     */
    public String[] getValues()
    {
        return thresholdEnums.getThresholdValues();
    }

    /**
     * Tests the ThresholdEnums control getThresholdValuesSimple
     * method. To test this method, simply click the getValuesSimple
     * button in Test View. Test View will refresh to display
     * a list of the enumeration values available for the threshold
     * attribute defined in the PriceSummary.xsd schema file.
     * That file is in the Schemas project of the SamplesApp application.
     *
     */
}
```

navWebServices.html Sample

```
* @common:operation
*/
public String[] getValuesSimple()
{
    return thresholdEnums.getThresholdValuesSimple();
}
}
```

TypeHierarchyPrinter.java Sample

This topic includes the source code for the TypeHierarchyPrinter.java Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/xmlBeans/schema/

Sample Source Code

```
package xmlBeans.schema;

import com.bea.xbean.tool.CommandLine;
import com.bea.xml.XmlObject;
import com.bea.xml.XmlOptions;
import com.bea.xml.XmlBeans;
import com.bea.xml.XmlException;
import com.bea.xml.SchemaTypeSystem;
import com.bea.xml.SchemaType;

import java.util.Collections;
import java.util.List;
import java.util.ArrayList;
import java.util.Collection;
import java.util.Iterator;
import java.util.HashMap;
import java.util.Map;
import java.util.Arrays;
import java.io.File;

/**
 * Provides a way to print a hierarchical list of the XML schema types
 * corresponding to Java types that result from compiling the schema.
 * The unique "signatures" used to represent the schema types in the hierarchy
 * are built from several characteristics, including:
 * <ul>
 * <li>The type's position in the schema (for example, is it global or local?).</li>
 * <li>Whether the type is a datatype (complex or simple) or an element or attribute.</li>
 * <li>Whether the type is derived by restriction, union, and so on.</li>
 * </ol>
 * There isn't yet a public standard for such signatures, but the style used here
 * is useful for seeing the hierarchy. Keep in mind that if a signature standard does
 * become adopted, that standard will likely be used by the SchemaType.toString()
 * method. See the WebLogic Workshop documentation for a more complete description
 * of the signature conventions used here.
 * <br/><br/>
 * This class is designed so that it may be used from either another
 * component (such as a web service) or from the command line.
 */
public class TypeHierarchyPrinter
{
    /**
     * Prints a hierarchical list of XML schema types represented by the
     * specified <em>typeSystem</em>.
     */
}
```


navWebServices.html Sample

```
public static String printHierarchy(SchemaTypeSystem typeSystem) throws Exception
{
    // A StringBuffer in which to build the response hierarchy.
    StringBuffer response = new StringBuffer();

    // A map for base SchemaType -> Collection of directly derived types.
    Map childTypes = new HashMap();

    // Traverse the type containment tree breadthfirst.
    List allSeenTypes = new ArrayList();
    // Document types are special types that hold the globally defined elements.
    allSeenTypes.addAll(Arrays.asList(typeSystem.documentTypes()));
    // Attribute types are special types that hold the globally defined attributes.
    allSeenTypes.addAll(Arrays.asList(typeSystem.attributeTypes()));
    /*
     * Global types are globally defined schema types -- such as complex or
     * simple types defined just beneath the schema's root in the hierarchy.
     */
    allSeenTypes.addAll(Arrays.asList(typeSystem.globalTypes()));

    /**
     * Loop through the types to examine,
     */
    for (int i = 0; i < allSeenTypes.size(); i++)
    {
        /**
         * A SchemaType object represents a compiled schema type. This
         * includes all three of the kinds added to allSeenTypes above.
         */
        SchemaType sType = (SchemaType)allSeenTypes.get(i);

        /*
         * Recurse through the nested anonymous types as well
         * (comment this line to skip nested types). Anonymous types
         * are defined inside other types.
         */
        allSeenTypes.addAll(Arrays.asList(sType.getAnonymousTypes()));

        /**
         * Don't examine nested document types, attribute types,
         * or the base type of anyType.
         */
        if (sType.isDocumentType() || sType.isAttributeType() || sType == XmlObject.type)
            continue;

        // Enter this type in the list of children of its base type.
        Collection children = (Collection)childTypes.get(sType.getBaseType());
        if (children == null)
        {
            children = new ArrayList();
            childTypes.put(sType.getBaseType(), children);

            /**
             * The first time a built-in type is seen add it, too,
             * to get a complete tree up to anyType, the root. A built-in type
             * is a schema type defined by the schema specification,
             * such as xs:string, xs:int, xs:anyType, and so on.
             */
            if (sType.getBaseType().isBuiltinType())
                allSeenTypes.add(sType.getBaseType());
        }
    }
}
```

navWebServices.html Sample

```

        children.add(sType);
    }

    /**
     * Print the hierarchy tree.
     */
    List typesToPrint = new ArrayList();

    // Add anyType, from which all others inherit.
    typesToPrint.add(XmlObject.type);

    // Create a buffer to hold the indentation spaces.
    StringBuffer spaces = new StringBuffer();

    /**
     * Loop through the list of types, adding the hierarchy branch
     * symbols and type "signatures".
     */
    while (!typesToPrint.isEmpty())
    {
        SchemaType sType = (SchemaType)typesToPrint.remove(typesToPrint.size() - 1);
        if (sType == null)
            spaces.setLength(Math.max(0, spaces.length() - 2));
        else
        {
            /**
             * The SchemaType.toString() method returns a String containing
             * the schema type "signature".
             */
            response.append(spaces + "+-" + sType.toString() + "\n");
            Collection children = (Collection)childTypes.get(sType);
            if (children != null && children.size() > 0)
            {
                spaces.append(typesToPrint.size() == 0 || typesToPrint.get(typesToPrint.size() - 1) != null ? "" : "\n");
                typesToPrint.add(null);
                typesToPrint.addAll(children);
            }
        }
    }
    return response.toString();
}

/**
 * Creates a schema type system from an array of XSD files. This method
 * is called when this class is used from the command line.
 */
public static SchemaTypeSystem typeSystemFromFiles(File[] schemaFiles)
{
    /**
     * A SchemaTypeSystem object will hold the type system created by
     * compiling the schema types.
     */
    SchemaTypeSystem typeSystem = null;

    List sdocs = new ArrayList();

    /**
     * Loop through the File array, parsing the contents of each schema
     * file into an XmlObject instance. These objects will be passed to
     * a method that compiles the schemas.
     */

```

navWebServices.html Sample

```

for (int i = 0; i < schemaFiles.length; i++)
{
    try
    {
        sdocs.add(XmlObject.Factory.parse(
            schemaFiles[i], (new XmlOptions()).setLoadLineNumbers()));
    }
    catch (Exception e)
    {
        System.err.println( schemaFiles[i] + " not loadable: " + e );
    }
}
XmlObject[] schemas = (XmlObject[])sdocs.toArray(new XmlObject[0]);

/**
 * Compile each schema. Use a Collection object to collect any
 * errors that arise during compilation.
 */
Collection compErrors = new ArrayList();
try
{
    /**
     * Compile the schemas, specifying also the built-in type system
     * to consult for already-compiled schema types which
     * may be linked while processing the given schemas.
     */
    typeSystem = XmlBeans.compileXsd(schemas,
        XmlBeans.getBuiltinTypeSystem(),
        new XmlOptions().setErrorListener(compErrors).setCompileDownloadUrls());
}
catch (XmlException e)
{
    System.out.println("Schema invalid");
    for (Iterator i = compErrors.iterator(); i.hasNext(); )
        System.out.println(i.next());
}
return typeSystem;
}

/**
 * A main method so that this class may be used from the command
 * line.
 */
public static void main(String[] args) throws Exception
{
    CommandLine cl = new CommandLine(args, Collections.EMPTY_SET);
    File[] schemaFiles = cl.GetFiles();

    SchemaTypeSystem typeSystem = typeSystemFromFiles(schemaFiles);
    String hierarchy = printHierarchy(typeSystem);
    System.out.println(hierarchy);
}
}

```

TypeHierarchyPrinterService.jws Sample

This topic includes the source code for the TypeHierarchyPrinterService.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/xmlBeans/schema/

Sample Source Code

```
package xmlBeans.schema;

import org.openuri.easypo.Customer;
import com.bea.xml.SchemaTypeSystem;

/**
 * @common:target-namespace namespace="http://workshop.bea.com/TypeHierarchyPrinterService"
 */

public class TypeHierarchyPrinterService implements com.bea.jws.WebService
{
    /**
     * Prints a hierarchical rendering of the schema types represented
     * by the schema type system available to this application.
     * A schema type system is a set of Java types corresponding to
     * XML schema types. When you compile schema into Java types,
     * those Java types (along with the built-in types they use or
     * inherit from) become part of a schema system.
     * <br/><br/>
     * The type system used by this sample is available to this code because
     * the types are included in a JAR file on this application's
     * classpath. These types are used by the various schema-related
     * samples in the SamplesApp application.
     *
     * @common:operation
     */
    public String printHierarchy() throws Exception
    {
        /**
         * Create a Customer instance simply to get the type system
         * it belongs to.
         */
        Customer customer = Customer.Factory.newInstance();

        /**
         * Get the type system from the Customer object.
         */
        SchemaTypeSystem typeSystem = customer.schemaType().getTypeSystem();

        /**
         * Pass the type system to a TypeHierarchyPrinter instance,
         * which will walk through it, creating a hierarchical rendering
         * of the schema types (including elements) defined in the
         * schemas from which the type system was created.
         */
    }
}
```

navWebServices.html Sample

```
    TypeHierarchyPrinter printer = new TypeHierarchyPrinter();  
    return printer.printHierarchy(typeSystem);  
  }  
}
```

XsdConfig.jws Sample

This topic includes the source code for the XsdConfig.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/xmlBeans/schema/

Sample Source Code

```
package xmlBeans.schema;

/**

Package and type names as they would be generated without
using an XSDCONFIG file.

import org.openuri.easypoLocal.PURCHORDERDocument;
import org.openuri.easypoLocal.PURCHORDERDocument.PURCHORDER;
import org.openuri.easypoLocal.PURCHORDERDocument.PURCHORDER.CUST;
import org.openuri.easypoLocal.PURCHORDERDocument.PURCHORDER.LINEITEM;
import org.openuri.easypoLocal.PURCHORDERDocument.PURCHORDER.SHIPPER;
import org.openuri.easypoLocal.NAMEDocument;
*/

import org.openuri.easypo.xsdconfig.PurchaseOrder;
import org.openuri.easypo.xsdconfig.PurchaseOrder.PurchaseOrder2;
import org.openuri.easypo.xsdconfig.PurchaseOrder.PurchaseOrder2.Customer;
import org.openuri.easypo.xsdconfig.PurchaseOrder.PurchaseOrder2.LineItem;
import org.openuri.easypo.xsdconfig.PurchaseOrder.PurchaseOrder2.Shipper;
import org.openuri.easypo.xsdconfig.Name;

/**
 * This web service illustrates how you can use an XSDCONFIG file
 * to guide naming translation when the schema compiler generates
 * an API corresponding to your schema. The schema for the XML
 * created by this code is EasyPOLocal.xsd, in the Schemas project
 * of the SamplesApp application. Without any configuration,
 * compiling that schema would result in the package and type names
 * shown in the commented import statements at the top of this
 * web service. <br/><br/>
 *
 * The EasyPOConfig.xsdconfig file included with the XSD file in
 * the Schemas project tells the schema compiler how to name and
 * package the types it generates. The XSDCONFIG file presents a
 * one-to-one mapping between schema element and proposed API name.
 * It also gives the compiler a package name to use instead of
 * the namespace URI. A name such as "PurchaseOrder2" above is
 * the schema compiler's effort to avoid a name conflict. Here, there
 * would be a conflict between the purchase order "document" type
 * that allows you to add a new PURCH_ORDER element to the document, and the
 * purchase order element that gives you access to its children. <br/><br/>
 *
 * Note that guiding the compiler-generated naming does not affect
 * names and namespaces for the underlying XML. The createPO
```

navWebServices.html Sample

```
* method of this web service returns the XML as it should be
* shaped according to the original schema.
*
* @common:target-namespace namespace="http://workshop.bea.com/XsdConfig"
*/
public class XsdConfig implements com.bea.jws.WebService
{
    /**
     * Creates purchase order XML from the information you enter
     * in the parameter boxes. Note that the returned XML appears
     * as defined by the schema even though the names for the API
     * used to construct it have been defined in an XSDCONFIG file.
     *
     * To test this method, enter values in the boxes and click
     * createPO.
     *
     * @common:operation
     */
    public PurchaseOrder createPO(String custName, String address,
        String description, int perUnitOunces, int quantity, double price)
    {
        PurchaseOrder poDoc =
            PurchaseOrder.Factory.newInstance();
        PurchaseOrder2 po = poDoc.addNewPurchaseOrder();

        Customer cust = po.addNewCustomer();
        cust.setName(custName);
        cust.setAddress(address);

        LineItem item = po.addNewLineItem();
        item.setDescription(description);
        item.setPerUnitOunces(perUnitOunces);
        item.setQuantity(quantity);
        item.setPrice(price);

        return poDoc;
    }
}
```

xquery Samples

This section contains source code for the following samples.

Samples Included in This Section

Employees.xml Sample

SelectPath.jws Sample

SimpleExpressions.jws Sample

Employees.xml Sample

This topic includes the source code for the Employees.xml Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/xmlBeans/xquery/

Sample Source Code

```
<xq:employees xmlns:xq="http://openuri.org/bean/samples/workshop/xmlBeans/xquery">
  <xq:employee>
    <xq:name>Fred Jones</xq:name>
    <xq:address location="home">
      <xq:street>900 Aurora Ave.</xq:street>
      <xq:city>Seattle</xq:city>
      <xq:state>WA</xq:state>
      <xq:zip>98115</xq:zip>
    </xq:address>
    <xq:address location="work">
      <xq:street>2011 152nd Avenue NE</xq:street>
      <xq:city>Redmond</xq:city>
      <xq:state>WA</xq:state>
      <xq:zip>98052</xq:zip>
    </xq:address>
    <xq:phone location="work">(425)555-5665</xq:phone>
    <xq:phone location="home">(206)555-5555</xq:phone>
    <xq:phone location="mobile">(206)555-4321</xq:phone>
  </xq:employee>
  <xq:employee>
    <xq:name>Sally Smith</xq:name>
    <xq:address location="home">
      <xq:street>1430 Oak Place</xq:street>
      <xq:city>Salem</xq:city>
      <xq:state>OR</xq:state>
      <xq:zip>97125</xq:zip>
    </xq:address>
    <xq:address location="work">
      <xq:street>765 Main St.</xq:street>
      <xq:city>Kaiser</xq:city>
      <xq:state>OR</xq:state>
      <xq:zip>97103</xq:zip>
    </xq:address>
    <xq:phone location="work">(503)555-3856</xq:phone>
    <xq:phone location="home">(503)555-6951</xq:phone>
    <xq:phone location="mobile">(503)555-5152</xq:phone>
  </xq:employee>
  <xq:employee>
    <xq:name>Gladys Kravitz</xq:name>
    <xq:address location="home">
      <xq:street>1313 Mockingbird Lane</xq:street>
      <xq:city>Seattle</xq:city>
      <xq:state>WA</xq:state>
      <xq:zip>98115</xq:zip>
    </xq:address>
```

navWebServices.html Sample

```
<xq:address location="work">
  <xq:street>2011 152nd Avenue NE</xq:street>
  <xq:city>Redmond</xq:city>
  <xq:state>WA</xq:state>
  <xq:zip>98052</xq:zip>
</xq:address>
<xq:phone location="work">(425)555-6897</xq:phone>
<xq:phone location="home">(206)555-6594</xq:phone>
<xq:phone location="mobile">(206)555-7894</xq:phone>
</xq:employee>
</xq:employees>
```

SelectPath.jws Sample

This topic includes the source code for the SelectPath.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/xmlBeans/xquery/

Sample Source Code

```
package xmlBeans.xquery;

import com.bea.xml.XmlCursor;
import com.bea.xml.XmlObject;

/**
 * A web service demonstrating how to use the selectPath
 * method to execute an XQuery expressions. Compare the code in this
 * web service with the code in the SimpleExpressions service. That
 * service uses the XmlCursor.execQuery method to execute expressions.
 * <br/><br/>
 *
 * You can call the selectPath method from either an XmlObject or XmlCursor
 * instance. Calling from XmlObject returns an XmlObject array. Calling
 * from XmlCursor returns void, and you use methods of the cursor to navigate
 * among returned "selections".
 *
 * @common:target-namespace namespace="http://workshop.bea.com/SelectPath"
 */
public class SelectPath implements com.bea.jws.WebService
{
    /**
     * Declare a namespace corresponding to the namespace declared
     * in the XML instance. The string here will be used as part of the
     * XPath expression to ensure that the query finds namespace-qualified
     * elements in the XML.
     */
    final static String m_namespaceDeclaration =
        "declare namespace xq='http://openuri.org/bea/samples/workshop/xmlBeans/xquery'";

    /**
     * Test this method by copying the entire contents of the Employees.xml file and
     * pasting them in place of the <AnyElement/> element. Click the
     * selectByState button to execute the following expression, which selects
     * any employee elements whose state child element has a value of "WA":
     * <br/><br/>
     *
     * &nbsp;$this/xq:employees/employee[address/state='WA']
     *
     * <br/><br/>
     *
     * The portion of this expression in square brackets -- [address/state='WA'] --
     * is known as a "predicate." The predicate filters items returned by the
     * expression to its immediate left. In other words, the first part of this
     * expression returns all of the employee elements, then the predicate

```

navWebServices.html Sample

```
* further filters those values to include only those whose address child element
* has a state child element with a value of "WA".
*
* This method illustrates how you can query from an XmlObject instance
* with the selectPath method. Results of the query (if any) are
* returned in an XmlObject array.
*
* @common:operation
*/
public XmlObject[] selectByState(XmlObject empDoc)
{
    /*
     * Create an XmlObject array to contain the results of the query.
     */
    XmlObject[] resultArray = null;

    /* A variable for the XQuery path expression. */
    String queryExpression = "$this/xq:employees/xq:employee[xq:address/xq:state='WA']";

    try
    {
        /*
         * Perform the query, combining the namespace declaration and the rest of query
         * expression.
         */
        resultArray = empDoc.selectPath(m_namespaceDeclaration + queryExpression);
    } catch (Exception e)
    {
        System.out.println(e.getLocalizedMessage());
    }
    /*
     * Return the array of results. Note that the returned results are contained
     * in an XmlObject element -- not an employee element.
     */
    return resultArray;
}

/**
 * Test this method by copying the entire contents of the Employees.xml file and
 * pasting them in place of the <AnyElement/> element. Click the
 * selectWorkPhonesAttr button to execute the following expression, which selects any
 * phone elements whose location attribute value is "work":
 * <br/><br/>
 *
 * &nbsp;$this/xq:employees/employee/phone[@location='work']
 *
 * <br/><br/>
 *
 * As with the selectByState method, this method
 * uses a path expression and predicate to specify the requested XML.
 *
 * This method illustrates how you can query from an XmlCursor instance
 * with the selectPath method. Unlike the selectByState example, the results of
 * this query (if any) are managed by the cursor from which selectPath
 * is called. Query results are available as "selections" in the cursor.
 * You navigate among them with methods such as XmlCursor.toNextSelection.
 *
 * @common:operation
 */
public XmlObject selectWorkPhonesAttr(XmlObject empDoc)
{
```

navWebServices.html Sample

```
// Add a cursor to the incoming XML.
XmlCursor empCursor = empDoc.newCursor();

// Create a new XmlObject instance in which to return the results of the query.
XmlObject resultXml = XmlObject.Factory.newInstance();

/*
 * Add a cursor to the new object; each of the returned results will be
 * copied to the location of this cursor. In other words, the XML
 * will be copied from cursor located in one XmlObject to a cursor located
 * in another.
 */
XmlCursor resultCursor = resultXml.newCursor();

/*
 * Move the result cursor to a location that's valid for receiving XML
 * copied from the set of results. When first inserted, the cursor is just
 * outside (or "left of") the STARTDOC token -- or outside the XML. This
 * call to toFirstToken moves it to a location between the STARTDOC and
 * ENDDOC, where XML can be inserted.
 */
resultCursor.toFirstContentToken();

// Create a variable with the query expression.
String queryExpression = "$this/xq:employees/xq:employee/xq:phone[@location='work']";

try
{
    // Execute the query.
    empCursor.selectPath(m_namespaceDeclaration + queryExpression);
    /*
     * Loop through the list of selections. For each selection,
     * copy its value to the location of a cursor in another
     * XmlObject.
     */
    while (empCursor.toNextSelection())
    {
        empCursor.copyXml(resultCursor);
    }
} catch (Exception e)
{
    System.out.println(e.getLocalizedMessage());
}

// Dispose of the query cursor.
empCursor.dispose();
// Return the XmlObject instance that now contains the results of the query.
return resultXml;
}

/**
 * Test this method by copying the entire contents of the Employees.xml file and
 * pasting them in place of the <AnyElement> element. Click the
 * selectEmpsByStateFLWR button to execute the following expression, which selects any
 * employee elements whose location state element value is "WA":
 * <br/><br/>
 *
 * &nbsp;for $e in $this/xq:employees/employee<br/>
 * &nbsp;&nbsp;let $s := $e/address/state<br/>
 * &nbsp;&nbsp;where $s = 'WA'<br/>
 * &nbsp;&nbsp;return $e<br/>
 * <br/><br/>
```

navWebServices.html Sample

```

*
* This method's expression illustrates a FLWR (pronounced "flower") expression.
* The letters in the acronym stand for "for... let... where... return". As you
* can probably imagine, a FLWR expression is a way to loop through XML,
* binding variables and evaluating based on the variables. This is very similar
* to loops in Java.
*
* Compare the results of this method with the results of the
* selectEmpsByStateFLWR method in the SimpleExpressions web service. That
* method uses the execQuery method to execute the expression, while this one
* uses selectPath.
*
* @common:operation
*/
public XmlObject[] selectEmpsByStateFLWR(XmlObject empDoc)
{
    XmlObject[] resultArray = null;
    /*
     * This expression loops through the employee elements, querying for the
     * state elements whose value is "WA", then returning the results.
     */
    String queryExpression =
        "for $e in $this/xq:employees/xq:employee " +
        "let $s := $e/xq:address/xq:state " +
        "where $s = 'WA' " +
        "return $e";

    try
    {
        // Execute the expression.
        resultArray = empDoc.selectPath(m_namespaceDeclaration + queryExpression);
    } catch (Exception e)
    {
        System.out.println(e.getLocalizedMessage());
    }
    return resultArray;
}

/**
 * Uses the selectPath method to execute the following
 * expression against the incoming XML:
 * <br/><br/>
 *
 * &nbsp;$this//xq:employee
 *
 * <br/><br/>
 * If results are returned, you can move the cursor to the first of
 * those results with the toNextSelection method. The method then
 * uses the following path expression to get the name element for
 * each employee element:
 * <br/><br/>
 *
 * &nbsp;$this//xq:name
 *
 * <br/><br/>
 * The name element values
 * are placed into a String array returned by the method.
 * <br/><br/>
 * To test this method, paste the contents of the Employees.xml
 * file between the getEmployees elements in the box on the
 * Test XML tab of Test View.
 *

```

navWebServices.html Sample

```
* @common:operation
*/
public String[] getEmployees(XmlObject employees)
{
    /**
     * Create a cursor with which to execute query expressions.
     * The cursor is inserted at the very beginning of the
     * incoming XML, then moved to the first element's start.
     */
    XmlCursor cursor = employees.newCursor();
    cursor.toNextToken();
    /**
     * Execute the path expression, qualifying it with the
     * namespace declaration.
     */
    cursor.selectPath(m_namespaceDeclaration + "$this//xq:employee");
    // Advance to the first selection in the list.
    cursor.toNextSelection();
    /**
     * Create an array to hold the name element values.
     */
    String[] names = new String[cursor.getSelectionCount()];
    /**
     * Loop through the selections, querying for the name
     * element, then placing the name element's value
     * in the String array.
     */
    for (int i = 0; i < cursor.getSelectionCount(); i++)
    {
        // An new cursor for the new query.
        XmlCursor nameCursor = cursor.newCursor();
        // Get the name element.
        nameCursor.selectPath(m_namespaceDeclaration +
            "$this/xq:name");
        // Advance to the first selection.
        nameCursor.toNextSelection();
        /**
         * Extract the name element's value and assign it
         * to the array.
         */
        names[i] = nameCursor.getTextValue();
        /**
         * Advance to the next selection in the original set
         * of results.
         */
        cursor.toNextSelection();
    }
    return names;
}
}
```

SimpleExpressions.jws Sample

This topic includes the source code for the SimpleExpressions.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/xmlBeans/xquery/

Sample Source Code

```
package xmlBeans.xquery;

import com.bea.xml.XmlCursor;
import com.bea.xml.XmlObject;

/**
 * This web service illustrates XQuery by showing a few simple expressions in use.
 * XQuery offers a variety of ways to operate on XML. It support simple path
 * expressions that are similar to XPath, as well as more complex expressions for
 * looping through XML.
 *
 * All of the XQuery expressions in the source code use the special word "$this".
 * This is not an XQuery word, but for XQuery expressions in WebLogic Workshop it
 * signifies "the current context in the XML instance document". Contrast this with
 * XQuery maps, which use $input to indicate "the start of the XML instance document".
 *
 * Also, notice that you execute an XQuery using an XmlCursor, and that you
 * should call the cursor's dispose method when you are finished with it. This
 * ensures that resources used by the cursor will be properly disposed of.
 *
 * For more information about XQuery, see http://www.w3.org/TR/xquery/.
 *
 * @common:target-namespace namespace="http://workshop.bea.com/SimpleExpressions"
 */
public class SimpleExpressions implements com.bea.jws.WebService
{
    /**
     * Declare a namespace for use with XQuery expressions. The syntax shown
     * in this string is the XQuery way to declare a particular namespace URI and
     * the prefix associated with it. The namespace declaration is needed because
     * the XML received by this web service is namespace-qualified. In order for
     * queries to work, that namespace must be taken into account.
     */
    final static String m_namespaceDeclaration =
        "declare namespace xq='http://openuri.org/bea/samples/workshop/xmlBeans/xquery'";

    /**
     * Test this method by copying the entire contents of the Employees.xml file and
     * pasting them in place of the <AnyElement> element. Click the
     * selectByState button to execute the following expression, which selects
     * any employee elements whose state child element has a value of "WA":
     * <br/><br/>
     *
     * &nbsp;$this/xq:employees/employee[address/state='WA']
     */
}
```


navWebServices.html Sample

```

*
* <br/><br/>
*
* The portion of this expression in square brackets -- [address/state='WA'] --
* is known as a "predicate." The predicate filters items returned by the
* expression to its immediate left. In other words, the first part of this
* expression returns all of the employee elements, then the predicate
* further filters those values.
*
* @common:operation
*/
public XmlObject selectByState(XmlObject empDoc)
{
    /*
     * Insert a cursor with which to execute the XQuery expression. Advance
     * the cursor to the first element in the document.
     */
    XmlCursor empCursor = empDoc.newCursor();
    XmlObject resultXML = null;

    /* A variable for the XQuery path expression. */
    String queryExpression = "$this/xq:employees/xq:employee[xq:address/xq:state='WA']";

    try{
        /*
         * Perform the query, combining the namespace declaration and the rest of query
         * expression.
         */
        XmlCursor resultCursor = empCursor.execQuery(m_namespaceDeclaration + queryExpression);

        /* Get an XmlObject instance containing the results of the query. */
        resultXML = resultCursor.getObject();

        /* Dispose of the result cursor. */
        resultCursor.dispose();

    }catch(Exception e){
        System.out.println(e.getLocalizedMessage());
    }
    /* Dispose of the document cursor. */
    empCursor.dispose();
    return resultXML;
}

/**
 * Test this method by copying the entire contents of the Employees.xml file and
 * pasting them in place of the <AnyElement/> element. Click the
 * selectWorkPhonesAttr button to execute the following expression, which selects any
 * phone elements whose location attribute value is "work":
 * <br/><br/>
 * &nbsp;$this/xq:employees/employee/phone[@location='work']
 * <br/><br/>
 *
 * As with the selectByState method, this method
 * uses a path expression and predicate to specify the requested XML.
 *
 * @common:operation
*/
public XmlObject selectWorkPhonesAttr(XmlObject empDoc)

```

navWebServices.html Sample

```
{
    XmlCursor empCursor = empDoc.newCursor();
    XmlObject resultXML = null;

    String queryExpression = "$this/xq:employees/xq:employee/xq:phone[@location='work']";

    try{
        XmlCursor resultCursor = empCursor.execQuery(m_namespaceDeclaration + queryExpressi
        resultXML = resultCursor.getObject();
        resultCursor.dispose();
    }catch(Exception e){
        System.out.println(e.getLocalizedMessage());
    }
    // Dispose of the query cursor.
    empCursor.dispose();
    return resultXML;
}

/**
 * Test this method by copying the entire contents of the Employees.xml file and
 * pasting them in place of the <AnyElement/> element. Click the
 * selectEmpsByStateFLWR button to execute the following expression, which selects any
 * employee elements whose location state element value is "WA":
 * <br/><br/>
 *
 * &nbsp;for $e in $this/xq:employees/employee<br/>
 * &nbsp;&nbsp;let $s := $e/address/state<br/>
 * &nbsp;&nbsp;where $s = 'WA'<br/>
 * &nbsp;&nbsp;return $e<br/>
 * <br/><br/>
 *
 * This method's expression illustrates a FLWR (pronounced "flower") expression.
 * The letters in the acronym stand for "for... let... where... return". As you
 * can probably imagine, a FLWR expression is a way to loop through XML,
 * binding variables and evaluating based on the variables. This is very similar
 * to loops in Java.
 *
 * @common:operation
 */
public XmlObject selectEmpsByStateFLWR(XmlObject empDoc)
{
    XmlCursor empCursor = empDoc.newCursor();
    empCursor.toNextSibling();

    XmlObject resultXML = null;

    /*
     * This expression loops through the employee elements, querying for the
     * state elements whose value is "WA", then returning the results.
     */
    String queryExpression =
        "for $e in $this/xq:employees/xq:employee " +
        "let $s := $e/xq:address/xq:state " +
        "where $s = 'WA' " +
        "return $e";

    try{
        XmlCursor resultCursor = empCursor.execQuery(m_namespaceDeclaration + queryExpressi
        resultXML = resultCursor.getObject();
        resultCursor.dispose();
    }catch(Exception e){
        System.out.println(e.getLocalizedMessage());
    }
}
```

navWebServices.html Sample

```

    }
    empCursor.dispose();
    return resultXML;
}

/**
 * Test this method by copying the entire contents of the Employees.xml file and
 * pasting them in place of the <AnyElement/> element. Click the
 * selectZipsNewDoc button to execute the following expression, which creates a
 * zip-list XML document that contains a list of zip element values:
 * <br/><br/>
 *
 * &nbsp;let $e := $this/xq:employees<br/>
 * &nbsp;&nbsp;return<br/>
 * &nbsp;&nbsp;<zip-list><br/>
 * &nbsp;&nbsp;&nbsp;{for $z in $e/employee/address/zip<br/>
 * &nbsp;&nbsp;&nbsp;&nbsp;return $z}<br/>
 * &nbsp;&nbsp;</zip-list><br/>
 * <br/><br/>
 *
 * This method illustrates what is known as an "element constructor."
 * That constructor in this case is the presence of the zip-list element
 * in the expression, along with that element's contents.
 * An element constructor is useful for creating XML of a different
 * shape from the results of your query expression. In an element
 * constructor, the expression contains XML tags that act as a kind
 * of template to hold query results.
 * <br/><br/>
 *
 * Here, the expression creates the zip-list element and inserts into
 * it the results of a for loop that returns zip elements in the XML.
 * The expression then returns the entire new element with zip-list
 * as its root.
 *
 * @common:operation
 */
public XmlObject selectZipsNewDoc(XmlObject empDoc)
{
    XmlCursor empCursor = empDoc.newCursor();
    empCursor.toNextSibling();

    XmlObject resultXML = null;

    String queryExpression =
        "let $e := $this/xq:employees " +
        "return " +
        "  <zip-list> " +
        "    {for $z in $e/xq:employee/xq:address/xq:zip " +
        "      return $z} " +
        "  </zip-list>";

    try{
        XmlCursor resultCursor = empCursor.execQuery(m_namespaceDeclaration + queryExpression);
        resultXML = resultCursor.getObject();
        resultCursor.dispose();
    }catch(Exception e){
        System.out.println(e.getLocalizedMessage());
    }
    empCursor.dispose();
    return resultXML;
}

```

}

xqueryMap Samples

This section contains source code for the following samples.

Samples Included in This Section

InputMapMultiple.jws Sample

InputMapMultipleTest.xml Sample

OutputMap.jws Sample

OutputScriptMap.jws Sample

Person.java Sample

PersonScript.jsx Sample

SimpleMap.jws Sample

InputMapMultiple.jws Sample

This topic includes the source code for the InputMapMultiple.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/xqueryMap/

Sample Source Code

```
package xqueryMap;

/**
 * This web service illustrates how you can use an XQuery map on a method whose
 * parameter values are arrays.
 *
 * @common:xmlns namespace="http://openuri.org/bea/samples/workshop/xmlmap/inputMultiple" prefix="x"
 * @common:target-namespace namespace="http://workshop.bea.com/InputMapMultiple"
 */
public class InputMapMultiple implements com.bea.jws.WebService
{
    /**
     * This method accepts three arrays as parameters and returns a simple float. In order
     * to map to the set of arrays, the XQuery map loops through the incoming XML message,
     * extracting values and inserting them into the template represented by the map. <br/><br/>
     *
     * To test this method, paste the contents of InputMapMultiple.xml into the Test XML
     * window provided in Test View, then click getTotalPrice.<br/><br/>
     *
     * In general, you may find that while the XQuery expression in the source code looks compl
     * it resembles programming conventions with which you're already familiar, such as variabl
     * loops, and return values.<br/><br/>
     *
     * Here are a few things to note about the XQuery in this map:<br/>
     * - An XQuery let clause binds variables (represented by $i, $n, and so on) to the
     * XML resulting from a path to an item element.<br/>
     * - for clauses evaluate their expressions and iterate over the items in the resulting
     * sequence, binding a variable to each item in turn. The bound variables are then used
     * to insert values into the template of the outgoing message.<br/>
     * - return clauses return the result of the expressions that follow them.<br/><br/>
     *
     * The return-xml XQuery map includes a simple path expression that finds the result of the
     * method and inserts it into the template for the outgoing message.<br/><br/>
     *
     * For the schema that defines the messages, see InputMapMultiple.xsd in the
     * Schemas project of the SamplesApp application.
     * <br/>
     *
     * @common:operation
     * @jws:parameter-xml schema-element="ns0:order" xquery::
     * declare namespace ns0="http://openuri.org/bea/samples/workshop/xmlmap/inputMultiple"
     * declare namespace ns1="http://workshop.bea.com/InputMapMultiple"
     *
     * let $i := $input/ns0:item
     *
     * return
```

navWebServices.html Sample

```

*      <ns1:getTotalPrice>
*          <ns1:nameArr>
*              {for $n in $i/ns0:name
*                  return
*                      <ns1:String>{data($n)}</ns1:String>}
*          </ns1:nameArr>
*          <ns1:amountArr>
*              {for $a in $i/ns0:amount
*                  return
*                      <ns1:int>{data($a)}</ns1:int>}
*          </ns1:amountArr>
*          <ns1:priceArr>
*              {for $p in $i/ns0:price
*                  return
*                      <ns1:float>{data($p)}</ns1:float>}
*          </ns1:priceArr>
*      </ns1:getTotalPrice>::
*  @jws:return-xml schema-element="ns0:totalPrice" xquery::
*  declare namespace ns0="http://workshop.bea.com/InputMapMultiple"
*  declare namespace ns1="http://openuri.org/bea/samples/workshop/xmlmap/inputMultiple"
*
*  <ns1:totalPrice>{data($input/ns0:getTotalPriceResult)}</ns1:totalPrice>
*  ::
*/
public float getTotalPrice(String [] nameArr, int [] amountArr, float [] priceArr)
{
    float totalPrice = 0.0f;

    // make sure list of items isn't empty
    if( nameArr != null )
    {
        // for each item, compute subtotal and add to total
        for (int i = 0; i < nameArr.length; i++)
        {
            totalPrice += amountArr[i] * priceArr[i];
        }
    }
    return totalPrice;
}
}

```

InputMapMultipleTest.xml Sample

This topic includes the source code for the InputMapMultipleTest.xml Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/xqueryMap/

Sample Source Code

```
<im:order xmlns:im="http://openuri.org/bea/samples/workshop/xmlmap/inputMultiple">
  <im:item>
    <im:name>Oranges</im:name>
    <im:amount>4</im:amount>
    <im:price>2.0</im:price>
  </im:item>

  <im:item>
    <im:name>Apples</im:name>
    <im:amount>12</im:amount>
    <im:price>3.6</im:price>
  </im:item>

  <im:item>
    <im:name>Pears</im:name>
    <im:amount>6</im:amount>
    <im:price>6.8</im:price>
  </im:item>

  <im:item>
    <im:name>Tangerines</im:name>
    <im:amount>13</im:amount>
    <im:price>6.31</im:price>
  </im:item>
</im:order>
```


OutputMap.jws Sample

This topic includes the source code for the OutputMap.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/xqueryMap/

Sample Source Code

```
package xqueryMap;

/**
 * This web service illustrates a simple return-XML XQuery map.
 *
 * @common:xmlns namespace="http://openuri.org/bea/samples/workshop/xmlmap/outputMap" prefix="n"
 * @common:target-namespace namespace="http://workshop.bea.com/OutputMap"
 */
public class OutputMap implements com.bea.jws.WebService
{
    public static class Person
    {
        public String fname;
        public String lname;
        public Person() {}
        public Person (String first, String last)
        {
            fname = first;
            lname = last;
        }
    }

    /**
     * This method uses a simple XQuery map to ensure that the outgoing message carrying
     * its return value conforms to a particular schema. The elements that make up the
     * map's template (elements such as ns1:EMPLOYEE) match those defined by the schema.
     * XQuery expressions (such as $input/ns0:HelloResult/ns0:fname) select values from
     * the default XML generated from the method's Java signature. At runtime,
     * WebLogic Server runtime components execute the XQuery expressions against
     * the default XML, then insert the resulting values into the template formed by
     * the map.<br/><br/>
     *
     * <strong>Be sure to use the Test XML tab -- not Text Form -- to test this method.</strong>
     * To test this method, replace the contents of the fname and lname elements that
     * Test View provides on the Test XML tab with your own values. In other words, you're
     * replacing the "string" default values.<br/><br/>
     *
     * For the schema that defines this method's outgoing message, see OutputMap.xsd in the
     * Schemas project of the SamplesApp application. The resulting XML itself will
     * be displayed in Test View when the method's results have returned.
     * <br/>
     *
     * @common:operation
     * @jws:return-xml xquery::
     *     declare namespace ns0 = "http://workshop.bea.com/OutputMap"
```

navWebServices.html Sample

```
*   declare namespace ns1 = "http://openuri.org/bea/samples/workshop/xmlmap/outputMap"
*
*   <ns1:EMPLOYEE>
*       <ns1:FIRSTNAME>{ data($input/ns0:HelloResult/ns0:fname) }</ns1:FIRSTNAME>
*       <ns1:LASTNAME>{ data($input/ns0:HelloResult/ns0:lname) }</ns1:LASTNAME>
*   </ns1:EMPLOYEE>
*   ::
*   schema-element="ns0:EMPLOYEE"
*/
public Person Hello (String fname, String lname)
{
    return new Person(fname, lname);
}
}
```

OutputScriptMap.jws Sample

This topic includes the source code for the OutputScriptMap.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/xqueryMap/

Sample Source Code

```
package xqueryMap;

import xqueryMap.Person;

/**
 * Demonstrates use of XQuery maps in combination with script
 * to do special formatting and processing when tranforming Java
 * objects to XML.
 * <br/><br/>
 * You can call script in a JSX file from an XQuery map to
 * incorporate logic that may not be possible in the XQuery itself.
 *
 * @common:xmlns namespace="http://openuri.org/bea/samples/workshop/xqueryMap/personScript" pre
 * @common:target-namespace namespace="http://workshop.bea.com/OutputScriptMap"
 */
public class OutputScriptMap implements com.bea.jws.WebService
{
    /**
     * Returns the XML response of this method as translated by
     * a JSX file called from an XQuery map. The JSX file contains
     * script that formats into another shape the XML that would
     * have otherwise been generated from the Person object this
     * method returns.
     * <br/><br/>
     * To test this method, enter any values in the fname and lname
     * boxes, then click the Hello button.
     *
     * @common:operation
     * @jws:return-xml schema-element="ns0:EMPLOYEE" xquery::
     * declare namespace ns0="http://workshop.bea.com/OutputScriptMap"
     * declare namespace ns1="http://openuri.org/bea/samples/workshop/xqueryMap/personScript"
     *
     * <ns1:EMPLOYEE>
     *     {xqueryMap.PersonScript.ConvertPerson($input)}
     * </ns1:EMPLOYEE>
     * ::
     */
    public Person Hello (String fname, String lname)
    {
        return new Person(fname, lname);
    }
}
```

Person.java Sample

This topic includes the source code for the Person.java Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/xqueryMap/

Sample Source Code

```
package xqueryMap;

/**
 * Represents information about a person as needed by the
 * OutputScriptMap.jws sample web service. This class is
 * used as the web service operation's return type, but
 * its contents are reshaped to a specific XML shape by
 * PersonScript.jsx.
 */
public class Person implements java.io.Serializable
{
    public String fname;
    public String lname;
    public Person ()
    {
//         fname = "Unknown";
//         lname = "User";
    }
    public Person (String first, String last)
    {
        fname = first;
        lname = last;
    }
}
```

PersonScript.jsx Sample

This topic includes the source code for the PersonScript.jsx Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/xqueryMap/

Sample Source Code

```
import xqueryMap.Person;

/*
 * Receives XML generated from the Hello method return type
 * in OutputScriptMap.jws. This function extracts the XML values
 * and places them into a slightly different XML shape (one whose
 * element names are capitalized). It returns that XML as the
 * actual return value of the Hello method.
 */
function ConvertPerson(person)
{
    /*
     * Declare the namespace that will be used when constructing
     * the XML below. The namespace declared here is the one used
     * by the XML received by this function.
     */
    ns = new Namespace("http://workshop.bea.com/OutputScriptMap");

    /*
     * Construct the XML that will be returned by this function.
     * The incoming XML's values are extracted using syntax such
     * as "person.ns::fname", where:
     *     "person" is an element name
     *     the dot indicates descent to a child element
     *     "ns" represents the namespace to which these elements belong
     *         (as declared above); "::" is a separator
     *     "fname" is a child of the person element.
     *
     * This XML is passed to the XQuery map that called this
     * script, where it is incorporated into the Hello method's
     * return value.
     */
    return <ns1:PERSON xmlns:ns1="http://openuri.org/bea/samples/workshop/xqueryMap/personScript">
        <ns1:FNAME>{person.ns::fname}</ns1:FNAME>
        <ns1:LNAME>{person.ns::lname}</ns1:LNAME>
    </ns1:PERSON>;
}
```

SimpleMap.jws Sample

This topic includes the source code for the SimpleMap.jws Sample.

Sample Location

This sample is located in the following directory in your WebLogic Workshop installation:

BEA_HOME/weblogic81/samples/workshop/SamplesApp/WebServices/xqueryMap/

Sample Source Code

```
package xqueryMap;

/**
 * A web service to illustrate a simple parameter-XML map that uses XQuery.
 *
 * @common:xmlns namespace="http://openuri.org/bea/samples/workshop/xmlmap/simpleMap" prefix="n"
 * @common:target-namespace namespace="http://workshop.bea.com/SimpleMap"
 */
public class SimpleMap implements com.bea.jws.WebService
{
    /**
     * This method uses a simple XQuery map to ensure that the parameter values in
     * the incoming XML message are correctly mapped to the parameters themselves.
     * The map itself (with its acceptPerson root element) reflects the method's
     * default XML schema -- a kind of default template for what an incoming message
     * should look like. However, the actual incoming message will be different; for
     * example, its root element is person, rather than acceptPerson.<br/><br/>
     *
     * To test this method, use the Test XML page in Test View. Looking at the
     * XML in the box provided there, replace the lastName and firstName values
     * with names of your own. Click acceptPerson to see the results of the test.<br/><br/>
     *
     * The XQuery code in the map's curly braces retrieves values from
     * the actual incoming XML message. Here, $input stands for the incoming
     * message's top element. What follows after it is a path to the part of the
     * XML whose value should be inserted where the XQuery expression is.<br/><br/>
     *
     * For the schema that defines the incoming message, see SimpleMap.xsd in the
     * Schemas project of the SamplesApp application.
     * <br/>
     *
     * @common:operation
     * @jws:parameter-xml xquery::
     *   declare namespace ns0 = "http://openuri.org/bea/samples/workshop/xmlmap/simpleMap"
     *   declare namespace ns1 = "http://workshop.bea.com/SimpleMap"
     *
     *   <ns1:acceptPerson>
     *     <ns1:lastName>{ data($input/ns0:lastname) }</ns1:lastName>
     *     <ns1:firstName>{ data($input/ns0:firstname) }</ns1:firstName>
     *   </ns1:acceptPerson>
     *   ::
     *   schema-element="ns0:person"
     */
    public String acceptPerson(String lastName, String firstName)
    {

```

navWebServices.html Sample

```
System.out.println(lastName + firstName);
lastName = lastName.trim();
firstName = firstName.trim();
return ("Received person: '" + firstName + " " + lastName + "'");
    }
}
```