BEA WebLogic Workshop™ Help

Version 8.1 SP4
December 2004

Copyright

Restricted Rights Legend

Trademarks or Service Marks

Table of Contents

JavaControlProject Samples

This section contains source code for the following samples.

Samples Included in This Section

docs Samples

verifyFunds Samples

docs Samples

This section contains source code for the following samples.

Samples Included in This Section

ControlAnnotations.xsd Sample

ControlAnnotations.xsd Sample

This topic inludes the source code for the ControlAnnotations.xsd Sample.

Sample Location

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

BEA_HOME/weblogic81/samples/workshop/SamplesApp/JavaControlProject/docs/

Sample Source Code

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 rel. 2 U (http://www.xmlspy.com) by Dave Read (BEA Systems, Inc.) --
<!-- edited with XML Spy v4.4 U (http://www.xmlspy.com) by David Read (private) -->
<xs:schema targetNamespace="http://www.bea.com/2003/03/controls/" xmlns="http://www.bea.com/200
  <xs:element name="jc-jar">
    <xs:annotation>
      <xs:documentation>Describes the contents of a control jar file.</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="control" maxOccurs="unbounded">
          <xs:annotation>
            <xs:documentation>The configuration of a specific control</xs:documentation>
          </xs:annotation>
          <xs:complexType>
            <xs:sequence minOccurs="0">
              <xs:element name="description" type="xs:string">
                <xs:annotation>
                  <xs:documentation>Description of the control.  Will have version attribute co
                </xs:annotation>
              </xs:element>
            </xs:sequence>
            <xs:attribute name="label" type="xs:string" use="required">
              <xs:annotation>
                <xs:documentation>The text displayed for this control in the palette (formerly
              </xs:annotation>
            </xs:attribute>
            <xs:attribute name="interface-class" type="xs:string" use="required">
              <xs:annotation>
                <xs:documentation>The fully qualified class name of the control interface.
</xs:documentation>
              </xs:annotation>
            </xs:attribute>
            <xs:attribute name="implementation-class" type="xs:string" use="optional">
              <xs:annotation>
                <xs:documentation>Fully qualified name of the control implementation class.  On
              </xs:annotation>
            </xs:attribute>
            <xs:attribute name="insert-wizard-class" type="xs:string" use="optional">
              <xs:annotation>
                <xs:documentation>Fully qualified classname for a user provided insert wizard.
              </xs:annotation>
            </xs:attribute>
            <xs:attribute name="version" type="xs:string" use="optional">
              <xs:annotation>
```

```
              <xs:documentation>A string representation of the control version.  The format a
            </xs:annotation>
          </xs:attribute>
          <xs:attribute name="icon-16" type="xs:string" use="optional">
            <xs:annotation>
              <xs:documentation>Path to an icon to be used in the control palette.  If no ico
            </xs:annotation>
          </xs:attribute>
          <xs:attribute name="icon-32" type="xs:string" use="optional"/>
          <xs:attribute name="group-priority" type="xs:int" use="optional">
            <xs:annotation>
              <xs:documentation>Optional group priority that can be used to organize the orde
            </xs:annotation>
          </xs:attribute>
          <xs:attribute name="group-name" type="ControlGroupNameType" use="optional"/>
          <xs:attribute name="palette-priority" type="PalettePriorityType" use="optional">
            <xs:annotation>
              <xs:documentation>Used as a hint to the IDE for ordering of controls within the
            </xs:annotation>
          </xs:attribute>
          <xs:attribute name="display-in-palette" type="xs:boolean" use="optional" default="t
            <xs:annotation>
              <xs:documentation>Defines whether this control will be displayed in the control
            </xs:annotation>
          </xs:attribute>
          <xs:attribute name="resource-file" type="xs:string" use="optional">
            <xs:annotation>
              <xs:documentation>Optional name of resource file to load when the control autho
1. label
2. version
3. description</xs:documentation>
            </xs:annotation>
          </xs:attribute>
          <xs:attribute name="requires-extension" type="xs:boolean" use="optional" default="f
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="default-group-name" type="ControlGroupNameType" use="optional">
      <xs:annotation>
        <xs:documentation>Default group name for controls in this jar file.  If a control has
</xs:documentation>
      </xs:annotation>
    </xs:attribute>
  </xs:complexType>
</xs:element>
<xs:element name="control-tags" type="ControlType">
  <xs:annotation>
    <xs:documentation>Describes the annotations for a specific control</xs:documentation>
  </xs:annotation>
</xs:element>
<xs:complexType name="ControlType">
  <xs:annotation>
    <xs:documentation>Structure of a control.  Note that this structure has been re-factored
  </xs:annotation>
  <xs:sequence>
    <xs:element name="control-tag" type="ControlTagType" minOccurs="0" maxOccurs="unbounded">
      <xs:annotation>
        <xs:documentation>A tag that applies to the control instance.</xs:documentation>
      </xs:annotation>
    </xs:element>
    <xs:element name="method-tag" type="MethodTagType" minOccurs="0" maxOccurs="unbounded">
```

```
      <xs:annotation>
        <xs:documentation>A tag that applies to a method within a JCX.  Javelin should verify
      </xs:annotation>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="editor-class" type="xs:string" use="optional">
    <xs:annotation>
      <xs:documentation>Fully qualified class name of a user provided class (e.g. property bu
    </xs:annotation>
  </xs:attribute>
  <xs:attribute name="control-validator" type="xs:string" use="optional"/>
  <xs:attribute name="method-validator" type="xs:string" use="optional"/>
</xs:complexType>
<xs:complexType name="BaseTagType">
  <xs:annotation>
    <xs:documentation>Structure of a tag that configures a control</xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="description" type="xs:string"/>
    <xs:element name="attribute" minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
        <xs:complexContent>
          <xs:extension base="TagAttributeType">
            <xs:attribute name="required" type="xs:boolean" use="required"/>
          </xs:extension>
        </xs:complexContent>
      </xs:complexType>
    </xs:element>
    <xs:element name="attribute-group" minOccurs="0" maxOccurs="unbounded">
      <xs:complexType>
        <xs:sequence>
          <xs:element name="attribute" type="TagAttributeType" minOccurs="2" maxOccurs="unbou
        </xs:sequence>
        <xs:attribute name="group-type" type="AttributeGroupType" use="required"/>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="name" type="xs:string" use="required">
    <xs:annotation>
      <xs:documentation>the name of the tag.  does not include prefix.</xs:documentation>
    </xs:annotation>
  </xs:attribute>
  <xs:attribute name="multiple" type="xs:boolean" use="optional" default="false">
    <xs:annotation>
      <xs:documentation>Whether the tag can have multiple occurrences.</xs:documentation>
    </xs:annotation>
  </xs:attribute>
  <xs:attribute name="validator-class" type="xs:string" use="optional">
    <xs:annotation>
      <xs:documentation>Fully-qualified name (as string) of an optional user provided class.
    </xs:annotation>
  </xs:attribute>
  <xs:attribute name="display-in-editor" type="xs:boolean" use="optional" default="true"/>
</xs:complexType>
<xs:simpleType name="PalettePriorityType">
  <xs:annotation>
    <xs:documentation>Range of priorities allowed for Palette Display</xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:integer">
    <xs:minInclusive value="1"/>
    <xs:maxInclusive value="10"/>
```

```
  </xs:restriction>
</xs:simpleType>
<xs:complexType name="TagAttributeType">
  <xs:sequence>
    <xs:element name="description" type="xs:string"/>
    <xs:element name="type">
      <xs:complexType>
        <xs:choice>
          <xs:element name="text">
            <xs:annotation>
              <xs:documentation>Simple text representation</xs:documentation>
            </xs:annotation>
            <xs:complexType>
              <xs:attribute name="max-length" type="xs:int" use="optional">
                <xs:annotation>
                  <xs:documentation>Max length of the text value.</xs:documentation>
                </xs:annotation>
              </xs:attribute>
              <xs:attribute name="long" type="xs:boolean" use="optional" default="false">
                <xs:annotation>
                  <xs:documentation>An indication for the IDE as to whether the text value sh
                </xs:annotation>
              </xs:attribute>
            </xs:complexType>
          </xs:element>
          <xs:element name="enumeration">
            <xs:annotation>
              <xs:documentation>A list of valid values</xs:documentation>
            </xs:annotation>
            <xs:complexType>
              <xs:sequence>
                <xs:element name="value" type="xs:string" maxOccurs="unbounded">
                  <xs:annotation>
                    <xs:documentation>A valid value as a string</xs:documentation>
                  </xs:annotation>
                </xs:element>
              </xs:sequence>
            </xs:complexType>
          </xs:element>
          <xs:element name="boolean">
            <xs:annotation>
              <xs:documentation>true/false</xs:documentation>
            </xs:annotation>
            <xs:complexType/>
          </xs:element>
          <xs:element name="decimal">
            <xs:annotation>
              <xs:documentation>non-integral value</xs:documentation>
            </xs:annotation>
            <xs:complexType>
              <xs:attribute name="places" type="xs:int" use="optional">
                <xs:annotation>
                  <xs:documentation>The number of decimal places</xs:documentation>
                </xs:annotation>
              </xs:attribute>
              <xs:attribute name="min-value" type="xs:decimal" use="optional">
                <xs:annotation>
                  <xs:documentation>The minimum value inclusive</xs:documentation>
                </xs:annotation>
              </xs:attribute>
              <xs:attribute name="max-value" type="xs:decimal" use="optional">
```

```
      <xs:annotation>
        <xs:documentation>The maximum value inclusive</xs:documentation>
      </xs:annotation>
    </xs:attribute>
  </xs:complexType>
</xs:element>
<xs:element name="integer">
  <xs:annotation>
    <xs:documentation>Integral value</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:attribute name="min-value" type="xs:long" use="optional">
      <xs:annotation>
        <xs:documentation>Minimum value inclusive</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="max-value" type="xs:long" use="optional">
      <xs:annotation>
        <xs:documentation>Maximum value inclusive</xs:documentation>
      </xs:annotation>
    </xs:attribute>
  </xs:complexType>
</xs:element>
<xs:element name="date">
  <xs:annotation>
    <xs:documentation>Date</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:attribute name="min-value" type="xs:date" use="optional">
      <xs:annotation>
        <xs:documentation>Minimum value inclusive</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="max-value" type="xs:date" use="optional">
      <xs:annotation>
        <xs:documentation>Maximum value inclusive</xs:documentation>
      </xs:annotation>
    </xs:attribute>
  </xs:complexType>
</xs:element>
<xs:element name="URI" type="xs:string"/>
<xs:element name="URN" type="xs:string"/>
<xs:element name="URL" type="xs:string"/>
<xs:element name="QNAME">
  <xs:annotation>
    <xs:documentation>An XML qualified name prefix:local part</xs:documentation>
  </xs:annotation>
  <xs:complexType/>
</xs:element>
<xs:element name="XML">
  <xs:annotation>
    <xs:documentation>Well-formed XML</xs:documentation>
  </xs:annotation>
  <xs:complexType/>
</xs:element>
<xs:element name="class-name" type="xs:string">
  <xs:annotation>
    <xs:documentation>Fully-qualified class name</xs:documentation>
  </xs:annotation>
</xs:element>
<xs:element name="class-names" type="xs:string">
```

```
            <xs:annotation>
              <xs:documentation>Space-separated list of fully qualified class names</xs:docum
            </xs:annotation>
          </xs:element>
          <xs:element name="file-path">
            <xs:annotation>
              <xs:documentation>Valid file path</xs:documentation>
            </xs:annotation>
          </xs:element>
          <xs:element name="custom">
            <xs:annotation>
              <xs:documentation>A custom (user defined) type</xs:documentation>
            </xs:annotation>
            <xs:complexType>
              <xs:attribute name="name" type="xs:string" use="required">
                <xs:annotation>
                  <xs:documentation>The name of the type</xs:documentation>
                </xs:annotation>
              </xs:attribute>
              <xs:attribute name="class-name" type="xs:string" use="required">
                <xs:annotation>
                  <xs:documentation>The fully qualified name of the class that implements an
                </xs:annotation>
              </xs:attribute>
            </xs:complexType>
          </xs:element>
        </xs:choice>
      </xs:complexType>
    </xs:element>
    <xs:element name="default-value" type="xs:string" minOccurs="0">
      <xs:annotation>
        <xs:documentation>Default value for the attribute.</xs:documentation>
      </xs:annotation>
    </xs:element>
  </xs:sequence>
  <xs:attribute name="name" type="xs:string" use="required">
    <xs:annotation>
      <xs:documentation>Attribute name
</xs:documentation>
    </xs:annotation>
  </xs:attribute>
</xs:complexType>
<xs:complexType name="ControlTagType">
  <xs:complexContent>
    <xs:extension base="BaseTagType">
      <xs:attribute name="allow-declaration-override" type="xs:boolean" use="optional" defaul
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<xs:complexType name="MethodTagType">
  <xs:complexContent>
    <xs:extension base="BaseTagType">
      <xs:attribute name="method-location" type="JbcxMethodType" use="optional" default="inte
        <xs:annotation>
          <xs:documentation>Defines the types of methods where this tag can exist.  Javelin s
</xs:documentation>
        </xs:annotation>
      </xs:attribute>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

```
  <xs:simpleType name="JbcxMethodType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="interface-method"/>
      <xs:enumeration value="callback-method"/>
      <xs:enumeration value="both"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="ControlGroupNameType">
    <xs:annotation>
      <xs:documentation>Optional group name that can be used to organize the control palette.</
    </xs:annotation>
    <xs:restriction base="xs:string"/>
  </xs:simpleType>
  <xs:simpleType name="AttributeGroupType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="at-most-one"/>
      <xs:enumeration value="exactly-one"/>
      <xs:enumeration value="at-least-one"/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

verifyFunds 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 inludes 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/JavaControlProject/verifyFunds/

Sample Source Code

```
package verifyFunds;

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 inludes 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/JavaControlProject/verifyFunds/

Sample Source Code

```
package verifyFunds;

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 ({poNumbe
     */
    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.
     *
```

```
     * @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 inludes 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/JavaControlProject/verifyFunds/poUtil/

Sample Source Code

```
package verifyFunds.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 inludes 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/JavaControlProject/verifyFunds/poUtil/

Sample Source Code

```
package verifyFunds.poUtil;

import java.util.StringTokenizer;

/**
 * Provides a number formatting utility function for the
 * VerifyFunds sample control.
 *
 * @jcs:jc-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 inludes 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/JavaControlProject/verifyFunds/

Sample Source Code

```
package verifyFunds;

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 inludes 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/JavaControlProject/verifyFunds/

Sample Source Code

```
package verifyFunds;

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. While this control
 * is identical to a control in the localControls folder of the
 * WebServices sample project, its placement here illustrates how
 * you can package a control in a control project for greater
 * portability. You can build this project (right-click JavaControlProject,
 * then click Build JavaControlProject) to generate a JavaControlProject.jar.
 * That file can in turn be used in multiple other applications. The JAR
 * file will be placed in the Libraries folder.
 *
 * @jcs:jc-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 verifyFunds.poUtil.POUtil poUtilJC;

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

```
/**
 * A Database control for accessing the item inventory.
 *
 * @common:control
 */
private verifyFunds.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 number. For the
 * the itemNumber and quantityRequested parameters, the value
 * matters because they 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</
 *
 * <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 number 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;
```

```
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:" + ac
                }
        return result;
    }

}
```