



BEA WebLogic Server™

WebLogic Builder Online Help

Release 7.0
Document Date: June 2002
Revised: June 28, 2002

Copyright

Copyright © 2002 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 Manager, BEA WebLogic Commerce Server, BEA WebLogic Enterprise, BEA WebLogic Enterprise Platform, 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.

WebLogic Builder Online Help

Part Number	Date	Software Version
N/A	June 28, 2002	BEA WebLogic Server Version 7.0

Contents

. **WebLogic Builder**

How WebLogic Builder Works.....	1-5
Recommended Uses for WebLogic Builder.....	1-6
Limitations of WebLogic Builder	1-6
Starting WebLogic Builder.....	1-7
Migrating a J2EE Module to WebLogic Server	1-10
Working with Web Applications.....	1-11
Adding a Servlet with Servlet Mapping and Security Constraints.....	1-11
Adding an ejb-ref/ejb-local-ref and Reference-Description.....	1-13
Adding a Resource-Reference.....	1-13
Adding a Listener Class	1-14
Adding a Filter with Filter Mapping	1-14
Defining a Match Map Class.....	1-15
Setting Welcome and Error Pages.....	1-15
Adding a Tag Library	1-16
Adding a Virtual Directory	1-16
Working with EJBs.....	1-17
Creating a Relationship Between 2.0 CMP Beans	1-17
Adding a CMP Field to an Entity Bean.....	1-19
Adding a Finder Method to an EJB.....	1-20
Specifying Optimistic Concurrency	1-21
Adding an ejb-reference Between Two Beans	1-22
Working with the J2EE Container.....	1-22
Ordering a Module	1-22
Setting Up EJB Caching.....	1-23
Choosing a Security Realm	1-23

WebLogic Builder User Interface.....	1-24
Menu Tasks	1-24
Deployment Descriptor Elements in WebLogic Builder.....	1-28
Porting and Deploying Smart Ticket with WebLogic Builder	1-1
Setting Up Applications and Environment	1-2
Converting and Tuning with WebLogic Builder	1-3
Generating Descriptors	1-3
Specifying JNDI Names and <context-root>	1-4
Administration Tasks.....	1-6
Configuring Data Sources	1-6
Deploying and Running	1-8
Running the Wireless Application.....	1-11
Summary.....	1-12
Related Information.....	1-12

WebLogic Builder

This document contains the following sections.

- [Starting WebLogic Builder](#)
- [Migrating a J2EE Module to WebLogic Server](#)
- [Working with Web Applications](#)
- [Working with EJBs](#)
- [Working with the J2EE Container](#)
- [WebLogic Builder User Interface](#)
- [Porting and Deploying Smart Ticket with WebLogic Builder](#)

How WebLogic Builder Works

WebLogic Builder is a visual environment for editing an application's deployment descriptor XML files. You can view descriptor files while you visually edit them in WebLogic Builder, and you won't need to make textual edits to the XML files.

What can I do using WebLogic Builder?

Begin by pointing WebLogic Builder at an application's compiled J2EE components (*.class files or modules that contain *.class files). If any of the deployment descriptor files needed for deployment on WebLogic Server are missing or defective, WebLogic Builder will offer to generate new or newly serviceable descriptor files.

Once deployment descriptor files exist, you can edit their elements and attributes using WebLogic Builder. You can, for example, add a tag library to a web application, or add a finder method to an EJB.

Test your application by using WebLogic Builder to deploy applications to a server.

Recommended Uses for WebLogic Builder

Use WebLogic Builder for the following tasks:

- Generate deployment descriptor files for a J2EE module
- Edit a module's deployment descriptor files
- View deployment descriptor files
- Compile and validate deployment descriptor files
- Deploy a module to a server

Limitations of WebLogic Builder

WebLogic Builder has some limitations.

- Cannot add new modules to an application's descriptor files
- Will notice changes to *.class files only if you close and reopen the module
- Support for generating descriptors for EJB 1.1 beans is not guaranteed; focus is on EJB 2.0
- Validates EJBs only
- Cannot automatically display the differences between a changed but unsaved descriptor file and original file
- Cannot perform batch descriptor update of XML element values
- Cannot remove components from a module's descriptor files
- If you make changes to descriptor files while they are opened in WebLogic Builder, Builder will not be aware of the changes
- No file management capabilities
- Generated XML representations of relations among entity beans are only accurate for one-to-one relations. For entity beans that already have descriptors, Builder does not refresh relations that have a "many" side.

Starting WebLogic Builder

Start Builder from the Start menu or from the command line.

From the Start menu, double-click WebLogic Builder under WebLogic 7.0.

In the command-line, use the following command:

for Windows:

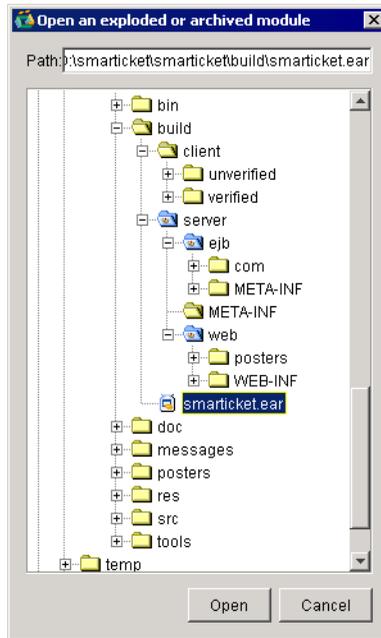
```
startWLBuilder.cmd
```

for Unix:

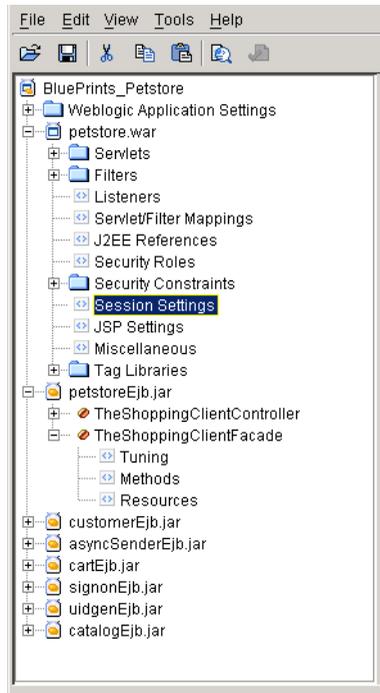
```
startWLBuilder.sh
```

This command sets your environment and starts WebLogic Builder.

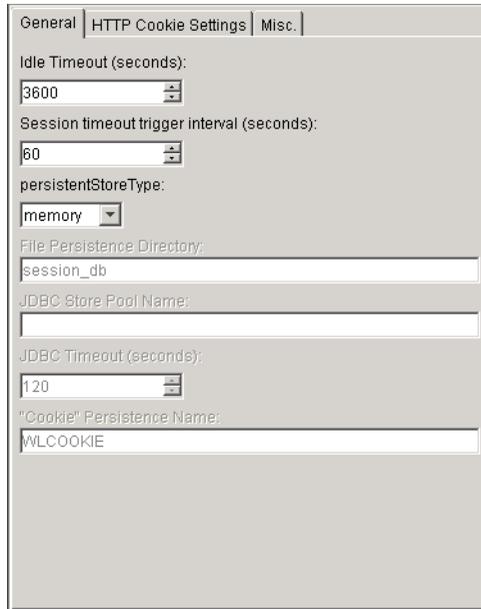
Open a module (a JAR or an EAR or a WAR or a J2EE module in exploded format) using the File menu's Open options.



When you open a module in WebLogic Builder, you see on the left a navigational tree view of the module's descriptor files, which you use to explore and select the components of the application.



On the right, you see tabbed panels with fields and other controls for editing the deployment descriptor elements of the module.



Use the navigational tree view on the left to select application components, and edit them in the corresponding tabs in the right-hand panel.

For more information about the interface, see [WebLogic Builder User Interface](#).

Migrating a J2EE Module to WebLogic Server

Migrate a module with no WebLogic Server deployment descriptors to WebLogic Server by opening the module using the File menu Open Archive or Open Directory.

WebLogic Builder checks that the module has all the deployment descriptor files required for successful deployment on WebLogic Server. If needed deployment descriptor files are missing, WebLogic Builder will offer to generate them for you. If you accept, WebLogic Builder will introspect the class files in your module and create appropriate deployment descriptor files.

WebLogic Builder will not overwrite your existing deployment descriptor files.

For more information about using WebLogic Builder to port applications to WebLogic Server, see [Porting and Deploying Smart Ticket with WebLogic Builder](#).

Working with Web Applications

See the following sections for information on Web Applications:

- [Adding a Servlet with Servlet Mapping and Security Constraints](#)
- [Adding an ejb-ref/ejb-local-ref and Reference-Description](#)
- [Adding a Resource-Reference](#)
- [Adding a Listener Class](#)
- [Adding a Filter with Filter Mapping](#)
- [Defining a Match Map Class](#)
- [Setting Welcome and Error Pages](#)
- [Adding a Tag Library](#)
- [Adding a Virtual Directory](#)

Adding a Servlet with Servlet Mapping and Security Constraints

This section describes how to add servlets to your Web Application's deployment descriptor files, and configure them with security roles, constraints, and assignments.

Adding a Servlet with URL Mapping

Use the following procedure to add a new servlet to the deployment descriptor file:

1. Under your Web Application's name in the navigational tree, select Servlets.
2. In the Servlets panel, select the servlet and click Add.
3. In the General tab, enter the Servlet Name and servlet class or JSP file.
4. Optionally, add URL mappings to the servlet in the URL mappings list by entering the URL pattern and clicking Add.
5. Click OK.

The servlet's name appears in the Servlet node in the navigational tree.

Adding Security Roles, Constraints, and Assignments

Add security constraints and assignments to security roles using the following procedures.

1. Under the Web Application node in the navigational tree, select Security Roles.
2. In the editing panel, click Add, enter security role names and descriptions, and click OK.
3. Under the Web Application node in the navigational tree, select Security Role Assignments.
4. In the editing panel, select a Role and click Add.
5. In the edit dialog, add the names of members of the Role.
6. Expand the Security Constraints node, and select a role node.
7. In the Resources/Pages tab, set the following:
 - Web Resource Name
 - URL patterns for
 - HTTP methods
 - allowed and disallowed roles
 - Transport Guarantee
 - Display Name
8. In the Roles tab, set the Roles for which the Resources/Pages settings are allowed.

Adding an ejb-ref/ejb-local-ref and Reference-Description

1. In the navigational tree, under the name of your Web Application, open the Servlets directory and select J2EE References.
2. In the J2EE References editing panel, select the EJB Refs tab and click Add.
3. Use the edit dialog to specify for the EJB the following, and then click OK:
 - EJB Reference Name
 - Link Name (optional)
 - EJB Type (Session or Entity)
 - Remote Interface
 - Home Interface
 - Run As (optional)
 - Description (optional)

WebLogic Builder writes the EJB reference to `web.xml` and lists it in the EJB Refs tab of the J2EE References editing panel.

Adding a Resource-Reference

Add a resource-reference to `web.xml` and then to `weblogic.xml` using the following procedure.

1. In the navigational tree, under the name of your Web Application, open the Servlets directory and select J2EE References.
2. In the J2EE editing panel, select the Resource Refs tab and click Add.
3. In the edit dialog, select the following:
 - Reference Type
 - Resource Sharing
 - Resource Authentication

Add a description and the Reference name, and click OK.

4. In the navigational tree, expand the WebLogic Settings node under Web Applications, select J2EE Links and click Add.
5. Select the Resource Reference that you added in step 3, enter its JNDI name for WebLogic, and click OK.

Adding a Listener Class

Add an event listener class to a Web Application using the following procedures:

1. In the navigational tree, under the name of your Web Application, select Miscellaneous.
2. In the Miscellaneous panel, select Listeners and click Add.
3. Enter the classname of the event and click OK.

Adding a Filter with Filter Mapping

1. Under the name of your Web Application in the navigational tree, select Filters.
2. In the Filters edit panel, click Add.
3. In the edit dialog, enter the display settings for the filter:
 - Display Name
 - Small Icon (must reside within the Web Application)
 - Description (optional)
 - Large Icon (also must reside within the Web Application)
4. Still in the dialog, select the Init Parameters tab and click Add.
5. Enter the filter's parameter name and value, and click OK.

Defining a Match Map Class

Define a match map to specify a class for URL pattern matching for your web application using the following procedure. The `url-match-map` element written using this procedure is located in `weblogic.xml`. See [url-match-map](#).

1. Under the name of your Web Application in the navigational tree, select the Miscellaneous node and then the Container Settings tab.
2. The Redirect content (determines the value for user-readable data used in a redirect) and Redirect content type (the servlet container uses this value to set the type on the response for internal redirects) fields do not persist values entered in them. Set these values in the text of `weblogic.xml`.
3. Check to specify whether redirects will use absolute URLs. If unchecked, the servlet container will not convert the relative url to the absolute URL in the location header in a redirect.
4. Specify whether to check authentication on forward. If checked, the request dispatcher will check authorization on forwarded requests.
5. Enter the name of a URL match-map class name for this Web Application.

Setting Welcome and Error Pages

Set the welcome and error pages for your Web Application using the following procedures:

1. Select your Web Application's name in the navigational tree.
2. In the edit panel for the Web Application, select Welcome Files.
3. Set the order for existing welcome files using the Move up, Move down buttons, or add new files by entering the filenames and clicking Add.
4. Select the Error Pages tab and click Add.
5. Enter the filename or browse to the file. Set HTTP Error Code or Exception Type, and click OK.

Adding a Tag Library

Add tag libraries using the following procedure:

1. Under your Web Application name in the navigational tree, select Tag Libraries.
2. In the Tag Libraries editing panel, click Add.
3. Enter the URI for the tag library.
4. Enter the location of the TLD or JAR file, or browse to it and select it.
5. Click OK.



Adding a Virtual Directory

Add a virtual directory to your Web Application using the following procedures:

1. In the navigational tree, under Web Application, WebLogic Settings, select VirtualDirectoryMappings and click Add.
2. Set a local directory path by entering it, and add its URL patterns by entering them in the bottom text field and clicking Add. Then click OK.

Working with EJBs

See the following sections for information about EJBs.

- [Creating a Relationship Between 2.0 CMP Beans](#)
- [Adding a CMP Field to an Entity Bean](#)
- [Adding a Finder Method to an EJB](#)
- [Adding an ejb-reference Between Two Beans](#)

Creating a Relationship Between 2.0 CMP Beans

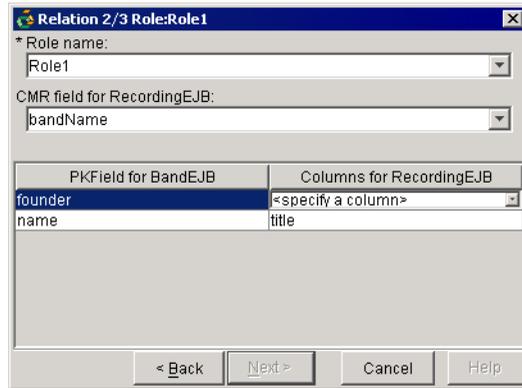
Create a relationship between two 2.0 CMP beans using the following procedure.

1. In the navigational tree under the EJB node, right-click the Relations node and select Add a relation... Note that if your module does not have CMP beans, WebLogic Builder will not display the Relations node.
2. In the Relations dialog, enter or select a name for the relation.

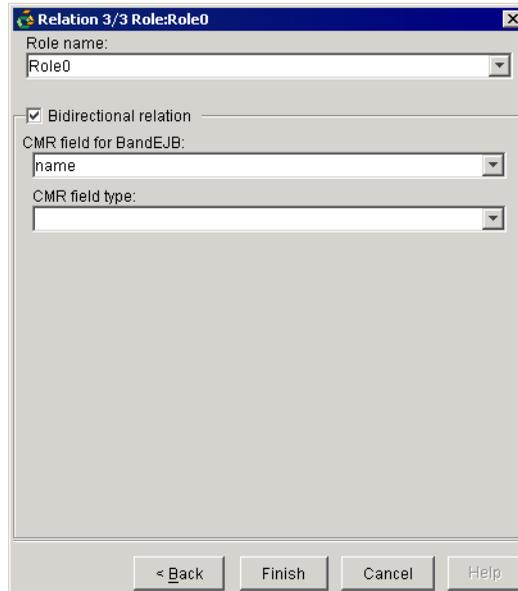


3. Set the Between option to One or Many for the first bean, and select the bean.
4. Set the And option for the second bean to the desired match with the first bean's Between option, and select the second bean and click Next.

5. In the second Relations dialog, select a Role name, a CMR (Container Managed Relationship) field for the second bean, and a primary key Field for the first bean and a column for the second bean.



6. Click Next.
7. In the third Relations dialog, select the role name and optionally set bidirectional relations (CMR field and field type).



8. Click Finish.

WebLogic Builder writes the relation to `ejb-jar.xml`, and an entry for the relation appears in the Relations node.

Adding a CMP Field to an Entity Bean

Add a Container Managed Persistence field to an entity bean using the following procedure.

1. In the navigational tree under EJB, expand an entity bean node and select the CMP Fields node.
2. In the CMP Fields dialog, select the field's name, which will correspond to a getter on your Bean class. For example, if you have `getFirstName()` on your Bean class, the name of the CMP field will be `firstName`.
3. Use the browse button to browse to a table name. If you are not connected to a server, the browse button will activate the Connect to a Server dialog.

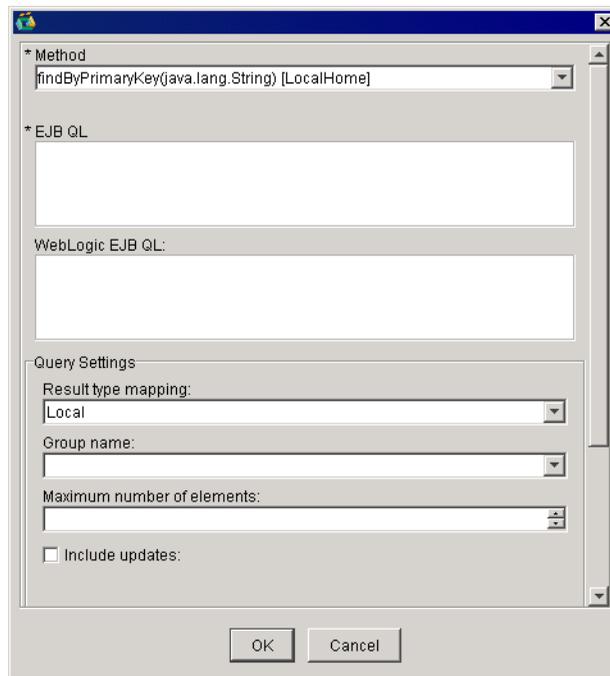
4. To select the column name, click the browse button and browse the table. Select a column and click OK.
5. Set the column type.
6. Click OK.

The new CMP field will appear in the navigational tree under the CMP node of the bean.

Adding a Finder Method to an EJB

To add a finder method to a bean, use the following procedure.

1. In the navigational tree, under the name of your entity bean, expand the bean and select Finders.
2. In the bean's Finder editing panel, click Add.



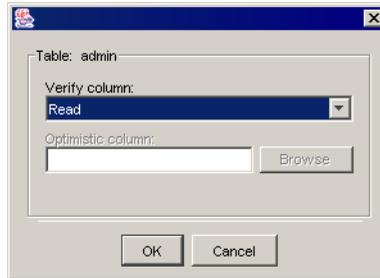
3. Select the method name, enter its properties, and click OK.

Specifying Optimistic Concurrency

You may want to set optimistic concurrency for your CMP entity beans when parallel transactions seem unlikely to conflict or when speed of response times is more important than certainty that transactions have not conflicted. WebLogic Builder's default setting is pessimistic concurrency. To set optimistic concurrency, do the following.

1. With an entity bean selected in the left navigational panel, select Tuning -> Cache. In the Concurrency strategy selection field, select Optimistic.

A dialog offers to let you select the Verify column and the Optimistic column to map the entity bean to a table.



2. Select Version or Timestamp in the Verify column to enable the Optimistic column field.
3. You can use the Browse button to connect to a server, browse a database, and select a column, or you can type the column name directly.

Adding an ejb-reference Between Two Beans

1. In the navigation tree, expand an EJB and select Resources.
2. In the EJB's Resources editing panel, select the EJB Refs or the EJB Local Refs tab and click Add.

Working with the J2EE Container

- [Ordering a Module](#)
- [Setting Up EJB Caching](#)
- [Choosing a Security Realm](#)

Ordering a Module

To set the deployment order of a module, use the following procedure.

1. In the navigational tree, select the module.
2. In the module's editing panel, select Deployment Order.
3. In the field that lists the module's components, select components and use the Move up and Move down buttons to reset their deployment order.

Setting Up EJB Caching

Set up EJB caching using the following procedure.

1. In the navigational tree, under the EJB node, expand a bean and select Tuning.
2. In the Tuning panel, set the following caching conditions.
 - Enter a the name of a concurrency strategy.
 - Check or leave unchecked the option to cache between transactions.
 - For a cache, set maximum number of beans in cache, idle time-out, and read time-out.
 - For cache reference, select the entity cache name and set the estimated bean size.

Choosing a Security Realm

Set a security realm for a module using the following procedure.

1. In the navigational tree, select the WebLogic Application Settings node.
2. In the WebLogic Application Settings editing panel, select the Security Realm tab.
3. In the Security Realm tab, enter the realm name.

WebLogic Builder User Interface

This section describes menu tasks and provides a key for locating deployment descriptor elements in the Builder interface.

[Menu Tasks](#)

[Deployment Descriptor Elements in WebLogic Builder](#)

Menu Tasks

- [Opening an Application](#)
- [Connecting to a Server](#)
- [Deploying](#)
- [Selecting a Compiler](#)
- [Closing an Application](#)
- [Saving an Application](#)
- [Validating an Application](#)
- [Generating Descriptors](#)
- [Removing a Component Descriptor](#)
- [Adding a New Descriptor Element](#)
- [Removing a Deployment Descriptor Element](#)
- [Viewing Deployment Descriptor XML Files](#)

Opening an Application

To open an archived or an exploded module, from the File menu, select Open. Browse to the archived module or to the directory that contains the exploded module, select it, and click Open.

Troubleshooting

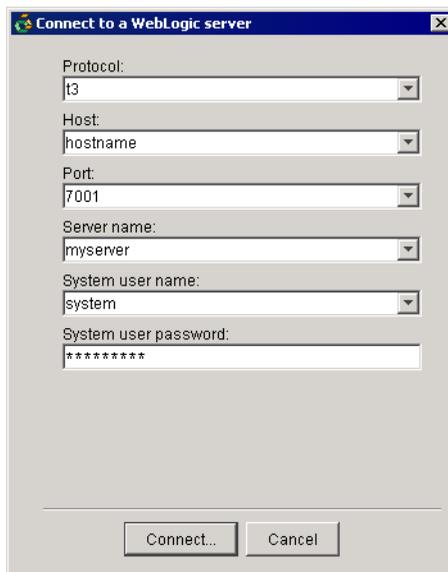
If you try to load a module that is not well formed, for example an EAR that has a nested JAR which is not referenced in the accompanying MANIFEST.MF file, WebLogic Builder may have trouble opening your module.

Connecting to a Server

Connect to a server to deploy your module for testing, or to connect your module to a data source.

From the tools menu, select Connect to Server..

Enter the connect information in the dialog, and click Connect.



The image shows a dialog box titled "Connect to a WebLogic server". It contains the following fields and controls:

- Protocol: t3 (dropdown menu)
- Host: hostname (dropdown menu)
- Port: 7001 (dropdown menu)
- Server name: myservers (dropdown menu)
- System user name: system (dropdown menu)
- System user password: ***** (text field)
- Buttons: Connect... and Cancel

Deploying

From the Tools menu, select Deploy Module. If you are not connected to a server, WebLogic Builder offers the Connect dialog.

Selecting a Compiler

1. From the Tools menu, select Options.
2. In the Options dialog, select EJBC Compiler.
3. Click Browse, and browse to the compiler. Select it and click Open.

Closing an Application

From the File menu, select Close.

Saving an Application

From the File menu, select Save.

All changes that you have made to deployment descriptor files in WebLogic Builder will be saved to your module.

Validating an Application

Validating does not save new changes to a module.

Select Validate from the Tools menu to validate your module.

Generating Descriptors

On opening a new module, WebLogic Builder asks you for permission to generate deployment descriptors for your opened module. When you accept, J2EE Application Builder creates the new descriptors and writes them to the appropriate location in the module.

Removing a Component Descriptor

Remove a component from the module by removing its associated descriptor element from the module, outside of WebLogic Builder.

Adding a New Descriptor Element

Add a new descriptor element to the module outside of WebLogic Builder.

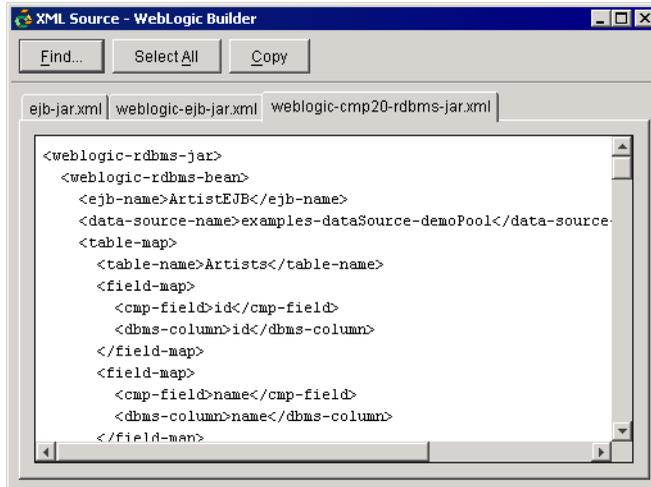
Removing a Deployment Descriptor Element

Remove files from module outside of WebLogic Builder.

Viewing Deployment Descriptor XML Files

View the XML files for the actively selected component using the following procedure. Note that these XML views are read-only.

1. From the View menu, select XML Source.
A tabbed XML viewer appears.
2. Use the tabs to select the XML file you wish to view.

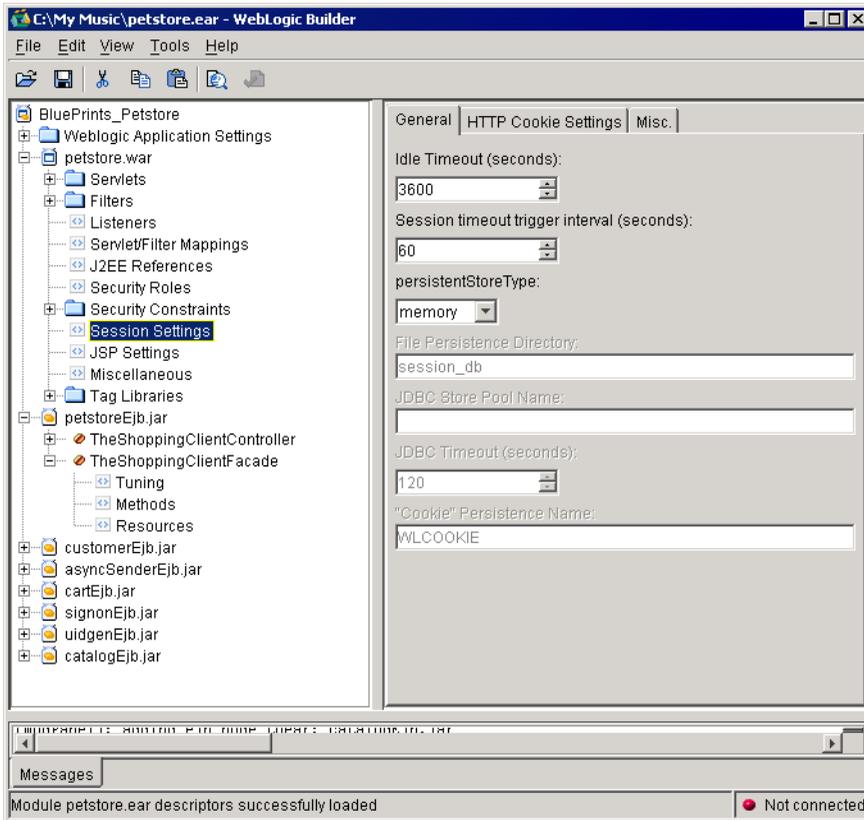


Deployment Descriptor Elements in WebLogic Builder

The sections listed below show the locations of deployment descriptor elements in WebLogic Builder.

- [weblogic.xml Elements in WebLogic Builder](#)
- [web.xml Elements in WebLogic Builder](#)
- [weblogic-application.xml Elements in WebLogic Builder](#)
- [ejb-jar.xml Elements in WebLogic Builder](#)
- [weblogic-ejb-jar.xml Elements in WebLogic Builder](#)
- [Tag Lib Elements in WebLogic Builder](#)
- [weblogic-cmp20-rdbms-jar.xml Elements in WebLogic Builder](#)

The file tree in the left panel contains nodes that group various deployment descriptor elements together in an intuitive manner. Click these nodes to navigate among the editing panels on the right, where you can make changes to the deployment descriptor elements.



weblogic.xml Elements in WebLogic Builder

The table below maps elements found in weblogic.xml to locations in WebLogic Builder's interface. See [weblogic.xml Deployment Descriptor Elements](#).

XML Element	WebLogic Builder Node → Tab
description	Web Application Node → Display

XML Element	WebLogic Builder Node → Tab
weblogic-version	
security-role-assignment: role-name, principal-name	Web Application Node→Security Roles
reference-descriptor: resource-description, res-ref-name, jndi-name, ejb-reference-description, ejb-ref-name, resource-env-description, res-env-ref-name	Web Application Node→J2EE References
session-descriptor: session-param, param-name: (CacheSize, ConsoleMainAttribute, CookieComment, CookieDomain, CookieMaxAgeSecs, CookieName, CookiePath, CookiesEnabled, IDLength, InvalidationIntervalSecs, JDBCConnectionTimeou tSecs, PersistentStoreCookieNa me, PersistentStoreDir, PersistentStorePool, PersistentStoreType, SwapIntervalSecs, TimeoutSecs, TrackingEnabled, URLRewritingEnabled)	Web Application Node→Session Settings

XML Element	WebLogic Builder Node → Tab
jsp-descriptor: jsp-param, param-name (compileCommand, compileFlags, compilerClass, compilerSupportsEncoding, defaultFilename, encoding, keepgenerated, noTryBlocks, packagePrefix, pageCheckSeconds, precompile, verbose, workingDir, debug)	Web Application Node→JSP Settings
container-descriptor	Web Application, Miscellaneous Node→Container Settings
charset-params	Web Application, Miscellaneous Node→IANA-Java Charset Mappings, and →Path-Charset Mappings
virtual-directory-mapping : (local-path, url-pattern)	Web Application, Miscellaneous Node→Virtual Directories
url-match-map	Web Application, Miscellaneous Node→Container Settings
security-permission	Web Application, Security Constraints

web.xml Elements in WebLogic Builder

The table below maps elements found in web.xml to locations in WebLogic Builder's interface. See [web.xml Deployment Descriptor Elements](#).

XML Elements and Attributes	WebLogic Builder Node → Tab
icon	Web Application Node→Display
display-name	Web Application Node→Display
description	Web Application Node→Display

XML Elements and Attributes	WebLogic Builder Node → Tab
distributable	not supported
context-param	Web Application Node→Context Params
filter: icon, filter-name, display-name, description, filter-class, init-param	Web Application, Filter Node→Filter
filter-mapping	Web Application, Servlet/Filter Mappings Node→Filter Mappings
listener	Web Application, Listeners→Listener Class
servlet: icon, servlet-name, display-name, description, (servlet-class jsp-file), init-param, load-on-startup, security-role-ref	Web Application, Servlets→Servlets
servlet-mapping: servlet-name, url-pattern	Web Application, Servlet/Filter Mappings Node→Servlet Mappings
session-config: session-timeout	Web Application, Session Settings→General
mime-mapping: extension, mime-type	Web Application→Mime Types
welcome-file-list	Web Application→Welcome Files
error-page: (error-code exception-type), location	Web Application→Error Pages

XML Elements and Attributes	WebLogic Builder Node → Tab
taglib: taglib-uri, taglib-location	Web Application, Tag Libraries→Tag Libraries
resource-env-ref: description, resource-env-ref-name, resource-env-ref-type	Web Application, J2EE References→Resource Env Refs
resource-ref: description, res-ref-name, res-type, res-auth, res-sharing-scope	Web Application, J2EE References→Resource Refs
security-constraint: display-name, web-resource-collection, auth-constraint, user-data-constraint	Web Application, Security Constraints→Resource Pages, Roles, SSL/Misc
login-config: auth-method, realm-name, form-login-config	Web Application→Login
security-role: description, role-name	Web Application, Security Roles→Role name, Description, Principal Names
security-role-ref: description, role-name, role-link	Web Application, Servlets, Servlet→Security Role Refs
env-entry: description, env-entry-name, env-entry-value, env-entry-type	Web Application, J2EE References→Env Entries

XML Elements and Attributes	WebLogic Builder Node → Tab
ejb-ref description, ejb-ref-name, ejb-ref-type, home, remote, ejb-link, run-as	Web Application, J2EE References→EJB Refs

weblogic-application.xml Elements in WebLogic Builder

The table below maps elements found in weblogic-application.xml to locations in WebLogic Builder’s interface. See [weblogic-application.xml Deployment Descriptor Elements](#).

XML Elements and Attributes	WebLogic Builder Node → Tab
weblogic-application	WebLogic Application Settings
ejb: entity-cache (entity-cache-name, (max-beans-in-cache max-cache-size), caching-strategy, start-mdbs-with-applicati on	WebLogic Application Settings, EJB Settings→EJB Settings
xml: parser-factory (saxparser-factory, document-builder-factory , transformer-factory), entity-mapping (entity-mapping-name, public-id, system-id, entity-uri, when-to-cache, cache-timeout-interval)	WebLogic Application Settings→XML Parser Factory, XML Entity Mappings

XML Elements and Attributes	WebLogic Builder Node → Tab
security: realm-name	WebLogic Application Settings→Security Realm
jdbc-connection-pool: data-source-name, connection-factory (factory-name, connection-properties), pool-params (size-params, xa-params, login-delay-seconds, leak-profiling-enabled, connection-check-params) , driver-params (statement, prepared-statement, row-prefetch-enabled, row-prefetch-size, stream-chunk-size), xa-params (debug-level, keep-conn-until-tx-compl ete-enabled, end-only-once-enabled, recover-only-once-enable d, tx-context-on-close-need ed, new-conn-for-commit-en abled, prepared-statement-cache -size, keep-logical-conn-open-o n-release, local-transaction-support ed, resource-health-monitori ng-enabled) acl-name	WebLogic Application Settings, JDBC Data Sources→General, Connection, Pool, XA Settings, Driver

ejb-jar.xml Elements in WebLogic Builder

The table below maps elements found in `ejb-jar.xml` to locations in WebLogic Builder's interface. See [WebLogic Server EJB Deployment Files](#).

XML Elements and Attributes	WebLogic Builder Node → Tab
<code>abstract-schema-name</code>	EJB → Advanced
<code>acknowledge-mode</code>	Message Driven Bean → Advanced
<code>security-role</code>	EJB → Security
<code>method-permission,</code>	EJB, Methods → Permissions
<code>container-transaction</code>	EJB, Methods → Transactions
<code>cascade-delete</code>	not supported
<code>cmp-field:</code> description, field-name	EJB, CMP Fields → CMP Fields
<code>cmp-version</code>	EJB → Persistence
<code>cmr-field:</code> description, cmr-field-name, cmr-field-type	EJB, Relations → Relation Wizard (right-click Relations)
<code>destination-type</code>	Message Driven Bean → General
<code>ejb-class</code>	EJB → Classes
<code>ejb-client-jar</code>	not supported
<code>ejb-link</code>	EJB → Resources
<code>ejb-local-ref:</code> description, ejb-ref-name, ejb-ref-type, local-home, local, ejb-link	EJB, Resources → EJB Local Refs

XML Elements and Attributes	WebLogic Builder Node → Tab
ejb-name	EJB→General
ejb-ql	EJB Application, Finders→Finders
ejb-ref: description, home, remote, ejb-link	EJB, Resources→EJB Refs
ejb-relation: description, ejb-relation-name, ejb-relationship-role	Relations→Relations Wizard (right-click on Relations)
ejb-relationship-role: description, ejb-relationship-role-name, multiplicity, relationship-role-source, cmr-field	Relations→Relations Wizard (right-click on Relations)
ejb-relationship-role-name	Relations→Relations Wizard (right-click on Relations)
ejb-class: home, remote, local-home, local	EJB→Classes
primkey-field	Entity Bean→Persistence
resource-env-ref: env-entry (description, env-entry-name, env-entry-type, env-entry-value)	EJB→Resources
field-name	Entity Bean, CMP Fields→CMP Fields

XML Elements and Attributes	WebLogic Builder Node → Tab
message-driven: ejb-name, ejb-class, message-driven-destination	Message Driven Bean→General, Classes
message-selector: acknowledge-mode, transaction-type	Message Driven Bean→Advanced
subscription-durability	Message Driven Bean→General
persistence-type	Entity Bean→Persistence
prim-key-class	Entity Bean→Persistence
primkey-field	Entity Bean→Persistence
query: description, query-method, result-type-mapping, ejb-ql	EJB, Finders→Finders
reentrant	EJB→Advanced
relationships: description, ejb-relation	EJB Application, Relations→Relations Wizard (right-click on Relations)
resource-env-ref: description, resource-env-ref-name, resource-env-ref-type	EJB, Resources→Environment
resource-ref: description, res-ref-name, res-type, res-auth, res-sharing-scope	EJB, Resources→Resource References
role-name	Enterprise Application→Security

XML Elements and Attributes	WebLogic Builder Node → Tab
session-type	Session Bean→General
session: ejb-name, home, remote, local-home, local, ejb-class, session-type, transaction-type,	Session Bean→General, Classes
session: env-entry, ejb-ref, ejb-local-ref, security-role-ref, security-identity, resource-ref, resource-env-ref	Session Bean, Resources→Environment, Resource References, EJB Refs, EJB Local Refs
subscription-durability	Message-Driven Bean→General
transaction-type	EJB, Methods→Transactions
trans-attribute	EJB, Methods→Transactions

weblogic-ejb-jar.xml Elements in WebLogic Builder

The table below maps elements found in weblogic-ejb-jar.xml to locations in WebLogic Builder's interface. See [weblogic-ejb-jar.xml Document Type Definitions](#).

XML Elements and Attributes	WebLogic Builder Node → Tab
cache-between-transactio ns	EJB, Tuning→Cache
concurrency-strategy	EJB, Tuning→Cache
connection-factory-jndi-n ame	Message-Driven Bean→Foreign JMS Provider

XML Elements and Attributes	WebLogic Builder Node → Tab
jms-polling-interval-seconds	Message-Driven Bean→Advanced
jms-client-id	Message-Driven Bean→Advanced
delay-updates-until-end-of-tx	EJB→Persistence
destination-jndi-name	Message-Driven Bean→General
ejb-reference-description: ejb-ref-name, jndi-name	EJB, Resources→EJB Refs
ejb-local-reference-description: ejb-ref-name, jndi-name	EJB, Resources→EJB Local Refs
enable-call-by-reference	For Session Bean: EJB→Advanced For Entity Bean: EJB→Persistence
enable-dynamic-queries	not supported
entity-cache: max-beans-in-cache, idle-timeout-seconds, read-timeout-seconds, concurrency-strategy, cache-between-transactions	EJB, Tuning→Cache Not supported for Entity EJB.
entity-cache-ref: entity-cache-name, concurrency-strategy, cache-between-transactions, estimated-bean-size	EJB, Tuning→Pool
entity-cache-name	not supported
estimated-bean-size	not supported

XML Elements and Attributes	WebLogic Builder Node → Tab
entity-clustering: home-is-clusterable, home-load-algorithm, home-call-router-class-na me	EJB, Tuning→Cluster
enable-dynamic-queries	not supported
finders-load-bean	EJB→Advanced
home-call-router-class-na me	Session and Entity Beans, Tuning→Cluster
home-is-clusterable	Session and Entity Beans, Tuning→Cluster
home-load-algorithm	Session and Entity Beans, Tuning→Cluster
idempotent-methods	EJB, Tuning→Cluster
idle-timeout-seconds	EJB, Tuning→Cache not supported for Stateful Session Bean
cache-type	not supported
initial-beans-in-free-pool	EJB, Tuning→Cluster
initial-context-factory	Message Driven Bean→Foreign JMS Provider
is-modified-method-nam e	not supported
isolation-level	Entity Bean, Methods→Transactions
jndi-name	Entity Bean→General
clients-on-same-server	not supported
local-jndi-name	EJB→General
max-beans-in-cache	EJB, Tuning→Cache
max-beans-in-free-pool	EJB, Tuning→Pool

XML Elements and Attributes	WebLogic Builder Node → Tab
message-driven-descriptor	not supported
persistence-use	not supported
pool: max-beans-in-free-pool, initial-beans-in-free-pool	EJB, Tuning→Pool
read-timeout-seconds	EJB, Tuning→Cache
replication-type	EJB, Tuning→Cluster
security-role-assignment: role-name, principal-name	EJB Application
stateful-session-clustering: home-is-clusterable, home-load-algorithm, home-call-router-class-name, replication-type	EJB, Tuning→Cluster
stateful-session-cache: max-beans-in-cache, idle-timeout-seconds	EJB, Tuning→Cache
stateless-bean-call-router-class-name	not supported
stateless-bean-is-clusterable	not supported
stateless-bean-load-algorithm	not supported
stateless-bean-methods-are-idempotent	not supported

XML Elements and Attributes	WebLogic Builder Node → Tab
stateless-clustering: home-is-clusterable, home-load-algorithm, home-call-router-class-name, stateless-bean-is-clusterable, stateless-bean-load-algorithm, stateless-bean-call-router-class-name, stateless-bean-methods-are-idempotent	not supported
stateless-session-descriptor: pool, stateless-clustering	not supported
transaction-isolation: isolation-level	not supported
trans-timeout-seconds	Entity Bean→Persistence
type-identifier	not supported
provider-url	Message Driven Bean→Foreign JMS Provider
invalidation-target: ejb-name	Entity Bean→Advanced

Tag Lib Elements in WebLogic Builder

The table below maps tag library elements to locations in WebLogic Builder's interface. See [Creating a Tag Library Descriptor](#).

XML Elements and Attributes	WebLogic Builder Node → Tab
taglib: tlib-version, jsp-version, short-name, uri, display-name, small-icon, large-icon, description, validator, listener	Web Application, Tag Libraries→Location, URI

weblogic-cmp20-rdbms-jar.xml Elements in WebLogic Builder

The table below maps elements found in weblogic-cmp20-rdbms-jar.xml to locations in the WebLogic Builder interface. See [WebLogic Server Container-Managed Persistence Services](#).

XML Elements and Attributes	WebLogic Builder Node → Tab
create-default-dbms-table	EJB→Application
delay-database-insert-until	Entity Bean→Advanced
automatic-key-generation	Entity Bean→Automatic Key Generation
field-group	not supported
table-map: table-name, field-map	EJB Application, Relations→Relation wizard (right-click Relations)
verify-columns, optimistic-column	not supported
check-exists-on-method	Entity Bean→Advanced
ejb-name	EJB→General

XML Elements and Attributes	WebLogic Builder Node → Tab
data-source-name	EJB→Persistence
table-name	EJB Application, Relations→Relation wizard (right-click Relations)
field-map: cmp-field, dbms-column, dbms-column-type	not supported
cmp-field	EJB Application, Relations→Relation wizard (right-click Relations)
dbms-column	EJB Application, Relations→Relation wizard (right-click Relations)
optimistic-column	not supported
dbms-column-type	EJB Application, CMP Fields, CMP→CMP
column-map: foreign-key-column, key-column	EJB Application, Relations→Relation wizard (right-click Relations)
weblogic-rdbms-relation: relation-name, table-name, weblogic-relationship-rol e, relationship-role-name	EJB Application, Relations→Relation wizard (right-click Relations)
relationship-role-map: foreign-key-table, primary-key-table, column-map	EJB Application, Relations→Relation wizard (right-click Relations)
group-name	EJB Application, Finders, Finder→Query Settings
cmr-field	EJB Application, Relations→Relation wizard (right-click Relations)

XML Elements and Attributes	WebLogic Builder Node → Tab
relationship-caching: caching-name, caching-element	not supported
caching-name	not supported
caching-element: cmr-field, group-name, caching-element	not supported
weblogic-query: query-method, weblogic-ql, group-name, max-elements, include-updates	EJB Application, Finders, Finder
sql-select-distinct	not supported
weblogic-ql	EJB Application, Finders, Finder
method-name	EJB Application, Finders, Finder
query-method	EJB Application, Finders, Finder
max-elements	EJB Application, Finders, Finder
include-updates	EJB Application, Finders, Finder
sql-select-distinct	EJB Application, Finders, Finder
automatic-key-generation : generator-type, generator-name, key-cache-size	EJB→Automatic Key Generation
generator-type	EJB→Automatic Key Generation
generator-name	EJB→Automatic Key Generation
key-cache-size	EJB→Automatic Key Generation

XML Elements and Attributes	WebLogic Builder Node → Tab
------------------------------------	------------------------------------

delay-database-insert-untill	EJB→Advanced
------------------------------	--------------

validate-db-schema-with	not supported
-------------------------	---------------

database-type	not supported
---------------	---------------

Porting and Deploying Smart Ticket with WebLogic Builder

This example shows a rapid deployment scenario for Sun's BluePrint wireless application, Smart Ticket. We use WebLogic Builder to generate and edit the WebLogic Server-specific deployment descriptor files and to deploy the application on WebLogic Server.

WebLogic Builder is a visual environment for generating and editing an application's deployment descriptor files. You can view descriptor files while you visually edit them in WebLogic Builder, and you won't need to make textual edits to the XML. See [WebLogic Builder at http://edocs.bea.com/wls/docs70/wlbuilder/index.html](http://edocs.bea.com/wls/docs70/wlbuilder/index.html).

This document contains following sections:

- [Setting Up Applications and Environment](#)
 - Download and install WebLogic Server 7.0
 - Download and install Smart Ticket
 - Download and install the Sun Wireless Toolkit
 - Set your environment
 - Build Smart Ticket
- [Converting and Tuning with WebLogic Builder](#)
 - Use WebLogic Builder to generate weblogic.xml and weblogic-ejb-jar.xml
 - Use WebLogic Builder to edit descriptors
- [Administration Tasks](#)
 - Start server (WebLogic Server's Examples server)
 - Use WebLogic Server Administration Console to configure datasources
 - Modify populate.bat script and SQL query to use the Pointbase sample database
- [Deploying and Running](#)

-
- Use WebLogic Builder to deploy Smart Ticket
 - Launch Smart Ticket.
 - Query and write to the datasources to create a user account and reserve movie tickets.

Setting Up Applications and Environment

You'll need the following to run this example.

- WebLogic Server 7.0
- Smart Ticket
- Sun's wireless toolkit, J2ME
- `smarticketPointBase.sql` (attached)

Here are the installation and build tasks and the path-settings you'll need.

1. Download and install WebLogic Server 7.0.

Download WebLogic Server 7.0 from <http://www.bea.com> and install it to a location we will hereafter call *WL_HOME*. By default, *WL_HOME* is `c:\bea\weblogic700`.

2. Download and install Smart Ticket 1.1.

Download the Smart Ticket demo application source code at : <http://developer.java.sun.com/developer/releases/smarticket/>. Extract it into a new directory on your machine. We will call this directory *SMARTICKET_HOME*.

3. Download and install J2ME to a location we will call *J2MEWTK_HOME* (default: `C:\J2mewtk`).

Download the Sun Wireless toolkit at: <http://java.sun.com/products/j2mewtoolkit/download.html>. Install the toolkit. During installation you will be prompted to select a JDK. You can select the JDK included in your *BEA_HOME*/jdk131 directory.

4. Set `J2MEWTK_HOME=C:\J2mewtk`, assuming `C:\J2mewtk` is where you installed J2ME. Note: if you do not set `J2MEWTK_HOME`, you will not be able to build the application.
5. Set your environment by running the `setExamplesEnv` script located in `WL_HOME\samples\server\config\examples`.
6. To `SMARTICKET_HOME\smarticket\localant.bat`, add “`%CLASSPATH%`” to the end of the `ANT_CLASSPATH` line.
7. Change to the `SMARTICKET_HOME\smarticket` directory and build Smart Ticket by running `localant.bat`. WebLogic Builder requires compiled `.class` files and cannot use `.java` files.

Converting and Tuning with WebLogic Builder

In this section, WebLogic Builder generates deployment descriptors, and we edit some of them without having to look at any XML.

Generating Descriptors

In this sequence, WebLogic Builder reads the existing descriptor files and introspects the application’s `.class` files to create the deployment descriptor files that help an application to run on WebLogic Server.

Note that WebLogic Builder will not overwrite the application’s existing deployment descriptor files.

1. Open WebLogic Builder from Start | Programs | BEA WebLogic Platform | WebLogic Server 7.0 | WebLogic Builder.

2. In WebLogic Builder's File | Open menu, navigate to *SMARTICKET_HOME* \build\server and click Open. A dialog asks: "Unable to locate deployment descriptors. Would you like deployment descriptors created for you?" Click Yes, and WebLogic Builder will introspect the Smart Ticket class files and generate *weblogic.xml* and *weblogic-ejb-jar.xml*.
3. Select File | Save and save the application in WebLogic Builder.
4. In WebLogic Builder, create *smarticket.ear* by selecting File | Archive and specifying *SMARTICKET_HOME* \bin\smarticket.ear.
5. Start the Examples server from: Start | Programs | BEA WebLogic Platform | WebLogic Server 7.0 | Server Tour and Examples | Launch Examples Server. The Examples Server launches a WebLogic Server Examples page.
6. Open the WebLogic Administration Console by navigating to <http://localhost:7001/console> (or by following the link from the WebLogic Server Examples page), and sign in using username *weblogic* and password *weblogic*.

Specifying JNDI Names and <context-root>

In this sequence, we use WebLogic Builder to:

- Set the web application's context path
 - Assign JNDI names to the web application's EJB references and resource references
 - Assign JNDI names to the EJB resource references
1. In WebLogic Builder, select the \web node's Context Path tab. In the Context Path text field, enter *SmarTicketApp*. Now the <context-root> element is set.
 2. Select Builder's EJB Refs panel in the J2EE Refs node, and specify the JNDI names for the EJB Refs as follows: *boo*

Reference Name	EJB Type	JNDI Name
<i>ejb/MovieInfo</i>	<i>Session</i>	<i>MovieInfo</i>

Reference Name	EJB Type	JNDI Name
ejb/TicketSales	Entity	TicketSales
ejb/Customer	Entity	Customer
ejb/LocaleInfo	Session	LocaleInfo

3. Select the Resource Refs panel in Builder's J2EE Refs node and set the Ref Name, EJB Type, and JNDI Name as in the table below. Set the Resource Authentication Type to Container.

Reference Name	Reference Type	JNDI Name
jdbc/MovieInfoDataSource	java.sql.DataSource	MovieInfoDataSource
jdbc/TicketSalesDataSource	java.sql.DataSource	TicketSalesDataSource
jdbc/CustomerDataSource	java.sql.DataSource	CustomerDataSource
jdbc/LocaleInfoDataSource	java.sql.DataSource	LocaleInfoDataSource

4. Select the Resource References panel from Builder's EJB Resources node and set JNDI names for EJB Resources as in the table below.

Reference Name	Resource Reference Type	JNDI Name
jdbc/MovieInfoDataSource	java.sql.DataSource	MovieInfoDataSource
jdbc/TicketSalesDataSource	java.sql.DataSource	TicketSalesDataSource
jdbc/CustomerDataSource	java.sql.DataSource	CustomerDataSource
jdbc/LocaleInfoDataSource	java.sql.DataSource	LocaleInfoDataSource

5. Use Builder's File | Save to save changes to the archive.

Administration Tasks

In this section we :

- Use the WebLogic Server Administration Console to configure the data sources
- Convert to the Pointbase example RDBMS.

Configuring Data Sources

Now we use the WebLogic Server administration console to configure the Data Sources for each of the four EJBs the Smart Ticket application uses.

1. If you haven't already started the WebLogic Examples Server and opened the Administration Console by pointing a browser to <http://localhost:7001/console>, do so now.
2. Select the JDBC node and click Tx Data Sources.
3. Select Configure a new JDBC Tx Data Source.
4. Enter a name for the data source. The first one is MyCustomerDataSource. Enter CustomerDataSource in the JNDI field. Enter demoPool as your Pool Name; this is the default connection pool that WebLogic examples use. Click Create.
5. Click the Targets Tab, then select the examplesServer in the Available column and click on the right arrow to target it. Click Apply.

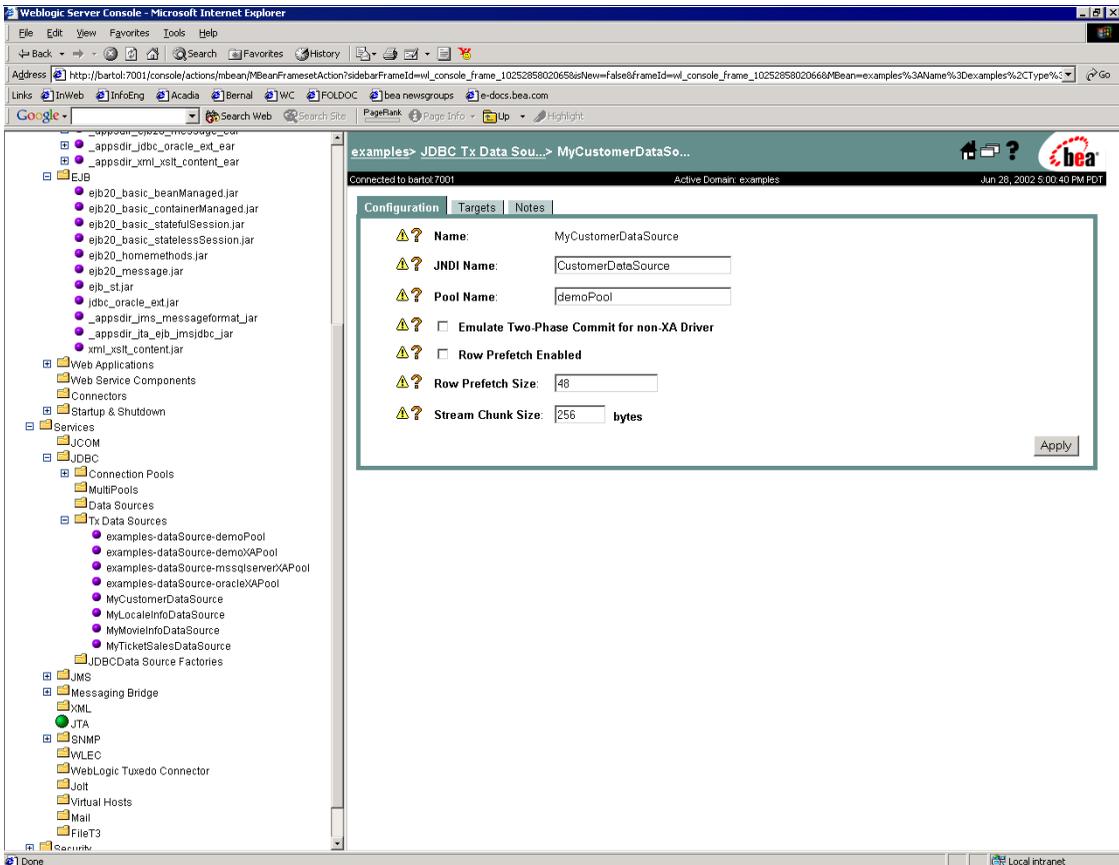


Figure 0-1 Setting up the Tx Data Sources in the Administration Console

- Repeat steps 4 and 5 for the other four data sources (MyMovieInfoDataSource, MyLocaleInfoDataSource, and MyTicketSalesDataSource).
- To replace Smart Ticket's Cloudscape database with Pointbase, the evaluation RDBMS included with WebLogic Server 7.0, just add the following to `SMARTICKET_HOME\smarticket\populate.bat`:

```
set POINTBASEHOME=%SAMPLES_HOME%\server\eval\pointbase
java utils.Schema
jdbc:pointbase:server://localhost/demo,database.home=%POINTBASE
```

```
HOME% com.pointbase.jdbc.jdbcUniversalDriver -u examples -p
examples -verbose ./src/smartticketPointBase.sql
```

8. We are also making the `smartticket.sql` Pointbase friendly by replacing it with a script that substitutes “int” data types with “integer.” Copy `smartticketPointBase.sql` from into `SMARTICKET_HOME\smartticket\src`.
9. Set up the database by running `populate.bat`.

Deploying and Running

Now we are ready to deploy and run Smart Ticket on WebLogic Server 7.0.

1. Connect to the server using the Connect to Server dialog in Weblogic Builder’s Tools menu.

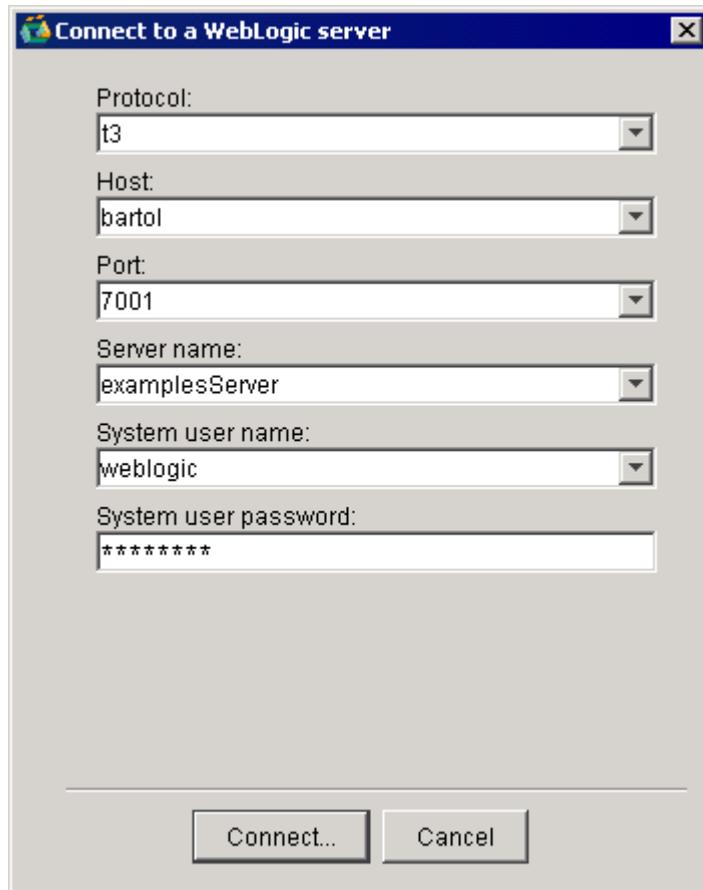
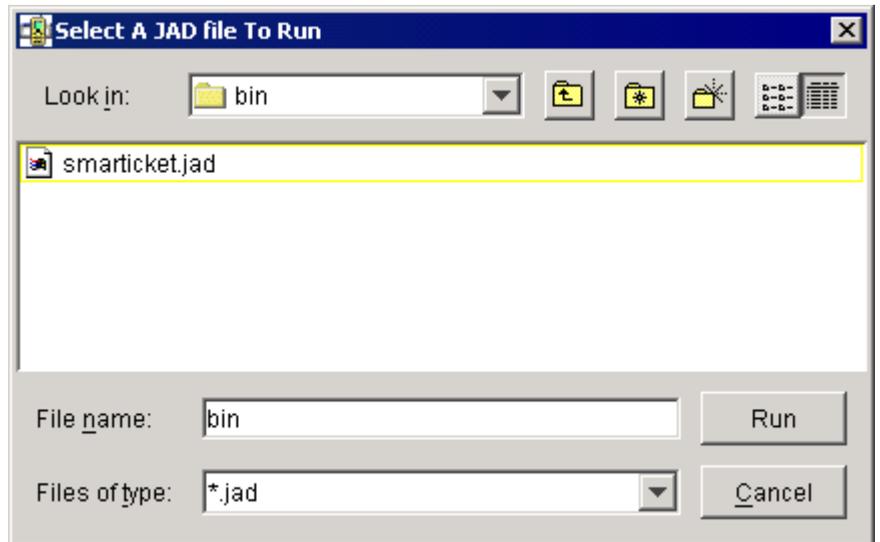


Figure 0-2 Connecting to the Examples Server

2. Deploy Smart Ticket by selecting Deploy Module from the WebLogic Builder tools menu.
3. Set the port in `SMARTICKET_HOME\bin\smarticket.jad` by replacing the given port number (8000) with the port number of the Examples server, which by default is 7001.
4. Start `SMARTICKET_HOME\bin\smarticket.jad` by double-clicking it or by selecting Start | J2ME Wireless Application | Run MIDP Application and selecting it:



5. You are running the Smart Ticket application on the Examples Server.



Running the Wireless Application

When creating a user account for the SmartTicket Application, enter in 95130 or 95054 for your zip code; also, your password must be 6 characters long. Try selecting the 'Poster' mode when you create an account; this will enable your phone or emulator to view a picture of the movie you choose to see.

There are several options available for using the emulator. Go into your J2ME Wireless Ticket and manipulate the default device and preferences.

If you have run the client Smart Ticket Application before and are redeploying the application on WebLogic, you will have to clear the database of the previous users information. To do so, run the J2ME Wireless Ticket Utility which is available from the Windows start menu. Click on 'Clean Database'. Then restart the Smart Ticket Client.

Summary

We downloaded the Sun application, built it, and use WebLogic Builder to generate the weblogic.xml and weblogic-ejb-jar.xml files required to run it on WebLogic Server. We then edited the application's elements slightly using WebLogic Builder, and used the WebLogic Server Administration Console to configure the application's datasources. We used WebLogic Builder to deploy the application, which we then launched and ran.

Related Information

See the procedure for porting Smart Ticket to WebLogic Server 7.0 without using WebLogic Builder: [Java Smart Ticket Demo 1.1](http://edocs.bea.com/wls/docs70/quickstart/smartticket.html) at <http://edocs.bea.com/wls/docs70/quickstart/smartticket.html>.

Index

A

- abstract-schema-name 1-36
- acknowledge-mode 1-36
- automatic-key-generation 1-44, 1-46
 - generator-name 1-46
 - generator-type 1-46
 - key-cache-size 1-46

C

- cache-between-transactions 1-39
- cache-timeout-interval 1-34
- caching-element 1-46
- caching-name 1-46
- caching-strategy 1-34
- cascade-delete 1-36
- charset-params 1-31
- check-exists-on-method 1-44
- clients 1-41
- clients-on-same-server 1-41
- cmp-field 1-36, 1-45
 - description 1-36
 - field-name 1-36
- cmp-version 1-36
- cmr-field 1-36
 - cmr-field-name 1-36
 - cmr-field-type 1-36
 - description 1-36
- column-map 1-45
- concurrency-strategy 1-39
- connection-factory 1-35

- connection-factory-jndi-name 1-39
- container-descriptor 1-31
- container-transaction 1-36
- context-param 1-32
- create-default-dbms-tables 1-44

D

- dbms-column 1-45
- dbms-column-type 1-45
- delay-database-insert-until 1-44
- delay-updates-until-end-of-tx 1-40
- deployment descriptor files, WebLogic
 - Builder will not overwrite existing 1-11
- description 1-31, 1-44
- destination-jndi-name 1-40
- destination-type 1-36
- display-name 1-31, 1-44
- distributable 1-32
- document-builder-factory 1-34

E

- ejb-class 1-36, 1-37
 - home 1-37
 - local 1-37
 - local-home 1-37
 - remote 1-37
- ejb-client-jar 1-36
- ejb-link 1-34, 1-36
- ejb-local-ref 1-36

- description 1-36
- ejb-link 1-36
- ejb-ref-name 1-36
- ejb-ref-type 1-36
- local 1-36
- local-home 1-36
- ejb-local-reference-description 1-40
- ejb-name 1-37, 1-44
- ejb-ql 1-37
- ejb-ref 1-34, 1-37
 - description 1-37
 - ejb-link 1-37
 - home 1-37
 - remote 1-37
- ejb-reference-description 1-40
 - ejb-ref-name 1-40
 - jndi-name 1-40
- ejb-ref-name 1-34
- ejb-ref-type 1-34
- ejb-relation 1-37
 - description 1-37
 - ejb-relation-name 1-37
 - ejb-relationship-role 1-37
- ejb-relationship-role 1-37
 - cmr-field 1-37
 - ejb-relationship-role-name 1-37
 - relationship-role-source 1-37
- ejb-relationship-role-name 1-37
- enable-call-by-reference 1-40
- enable-dynamic-queries 1-40, 1-41
- entity-cache 1-34, 1-40
 - cache-between-transactions 1-40
 - concurrency-strategy 1-40
 - idle-timeout-seconds 1-40
 - max-beans-in-cache 1-40
 - read-timeout-seconds 1-40
- entity-cache-name 1-34
- entity-clustering 1-41
 - home-call-router-class-name 1-41
 - home-is-clusterable 1-41

- home-load-algorithm 1-41
- entity-mapping 1-34
- entity-mapping-name 1-34
- entity-uri 1-34
- env-entry 1-33
- env-entry-name 1-33
- env-entry-type 1-33
- env-entry-value 1-33
- error-page
 - error-code|exception-type 1-32
 - location 1-32

F

- field-group 1-44
- field-map 1-44, 1-45
 - cmp-field 1-45
 - dbms-column 1-45
 - dbms-column-type 1-45
- filter 1-32
 - display-name 1-32
 - filter-name 1-32
 - icon 1-32
- filter-class 1-32
- filter-mapping 1-32
- finders-load-bean 1-41
- foreign-key-column 1-45
- foreign-key-table 1-45

G

- generator-name 1-46
- generator-type 1-46

H

- home 1-39
- home-call-router-class-name 1-41, 1-42
- home-is-clusterable 1-41, 1-42
- home-load-algorithm 1-41, 1-42

I

- icon 1-31
- idempotent-methods 1-41
- idle-timeout-seconds 1-41, 1-42
- include-updates 1-46
- initial-beans-in-free-pool 1-41, 1-42
- initial-context-factory 1-41
- init-param 1-32
- invalidation-target 1-43
 - ejb-name 1-43
- isolation-level 1-41, 1-43

J

- jdbc-connection-pool 1-35
 - acl-name 1-35
 - connection-factory
 - connection-properties 1-35
 - factory-name 1-35
 - data-source-name 1-35
 - driver-params 1-35
 - prepared-statement 1-35
 - row-prefetch-enabled 1-35
 - row-prefetch-size 1-35
 - statement 1-35
 - stream-chunk-size 1-35
 - pool-params 1-35
 - connection-check-params 1-35
 - leak-profiling-enabled 1-35
 - login-delay-seconds 1-35
 - size-params 1-35
 - xa-params 1-35
 - xa-params 1-35
 - debug-level 1-35
 - end-only-once-enabled 1-35
 - keep-conn-until-tx-complete-enabled 1-35
 - keep-logical-conn-open-on-release 1-35
 - local-transaction-supported 1-35
 - new-conn-for-commit-enabled 1-35

- prepared-statement-cache-size 1-35
- recover-only-once-enabled 1-35
- resource-health-monitoring-enabled 1-35
 - tx-context-on-close-needed 1-35
- jms-client-id 1-40
- jms-polling-interval-seconds 1-40
- jndi-name 1-41
- jsp-descriptor 1-31
 - jsp-param 1-31
 - param-name 1-31
 - compileCommand 1-31
 - compileFlags 1-31
 - compilerClass 1-31
 - compilerSupportsEncoding 1-31
 - debug 1-31
 - defaultFilename 1-31
 - keepgenerated 1-31
 - noTryBlocks 1-31
 - packagePrefix 1-31
 - pageCheckSeconds 1-31
 - precompile 1-31
 - verbose 1-31
 - workingDir 1-31
- jsp-file 1-32
- jsp-version 1-44

K

- key-cache-size 1-46
- key-column 1-45

L

- large-icon 1-44
- listener 1-32, 1-44
- load-on-startup 1-32
- local-jndi-name 1-41
- login-config 1-33
 - auth-method 1-33
 - form-login-config 1-33

realm-name 1-33

M

max-beans-in-cache 1-34, 1-41, 1-42
max-beans-in-free-pool 1-41, 1-42
max-cache-size 1-34
max-elements 1-46
message-driven 1-38
 ejb-name 1-38
message-driven-descriptor 1-42
message-driven-destination 1-38
message-selector 1-38
 acknowledge-mode 1-38
method-permission, 1-36
mime-mapping 1-32
 extension 1-32
 mime-type 1-32
multiplicity 1-37

O

optimistic-column 1-45

P

parser-factory 1-34
persistence-type 1-38
persistence-use 1-42
pool 1-42
primary-key-table 1-45
prim-key-class 1-38
primkey-field 1-37, 1-38
principal-name 1-42
provider-url 1-43
public-id 1-34

Q

query 1-38
 description 1-38
 ejb-ql 1-38

result-type-mapping 1-38
query-method 1-38, 1-46

R

read-timeout-seconds 1-42
realm-name 1-35
reentrant 1-38
reference-descriptor 1-30
 ejb-reference-description 1-30
 ejb-ref-name 1-30
 jndi-name 1-30
 res-env-ref-name 1-30
 resource-description 1-30
 resource-env-description 1-30
 res-ref-name 1-30
relation-name 1-45
relationship-caching 1-46
relationship-role-map 1-45
relationship-role-name 1-45
relationships 1-38
 description 1-38
 ejb-relation 1-38
replication-type 1-42
resource-env-ref 1-33, 1-37, 1-38
 description 1-38
 env-entry 1-37
 env-entry-name 1-37
 env-entry-type 1-37
 env-entry-value 1-37
 resource-env-ref-name 1-33, 1-38
 resource-env-ref-type 1-33, 1-38
resource-ref 1-33, 1-38
 description 1-38
 res-auth 1-33, 1-38
 res-ref-name 1-33, 1-38
 res-sharing-scope 1-33, 1-38
 res-type 1-33, 1-38
role-name 1-38, 1-42
run-as 1-34

S

- saxparser-factory 1-34
- security-constraint 1-33
 - auth-constraint 1-33
 - display-name 1-33
 - user-data-constraint 1-33
 - web-resource-collection 1-33
- security-permission 1-31
- security-role 1-33, 1-36
 - description 1-33
 - role-name 1-33
- security-role-assignment 1-30, 1-42
 - principal-name 1-30
 - role-name 1-30
- security-role-ref 1-32, 1-33
 - description 1-33
 - role-link 1-33
 - role-name 1-33
- servlet 1-32
- servlet-class 1-32
- servlet-mapping 1-32
 - servlet-name 1-32
 - url-pattern 1-32
- servlet-name 1-32
- session 1-39
 - ejb-class 1-39
 - ejb-local-ref 1-39
 - ejb-name 1-39
 - ejb-ref 1-39
 - env-entry 1-39
 - local 1-39
 - local-home 1-39
 - remote 1-39
 - resource-env-ref 1-39
 - resource-ref 1-39
 - security-identity 1-39
 - security-role-ref 1-39
 - session-type 1-39
 - transaction-type 1-39
- session-config 1-32

- session-descriptor 1-30
 - param-name 1-30
 - CacheSize 1-30
 - ConsoleMainAttribute 1-30
 - CookieComment 1-30
 - CookieDomain 1-30
 - CookieMaxAgeSecs 1-30
 - CookieName 1-30
 - CookiePath 1-30
 - CookiesEnabled 1-30
 - IDLength 1-30
 - InvalidationIntervalSecs 1-30
 - JDBCConnectionTimeoutSecs 1-30
 - PersistentStoreCookieName 1-30
 - PersistentStoreDir 1-30
 - PersistentStorePool 1-30
 - PersistentStoreType 1-30
 - SwapIntervalSecs 1-30
 - TimeoutSecs 1-30
 - TrackingEnabled 1-30
 - URLRewritingEnabled 1-30
 - session-param 1-30
- session-timeout 1-32
- session-type 1-39
- short-name 1-44
- small-icon 1-44
- sql-select-distinct 1-46
- start-mdbs-with-application 1-34
- stateful-session-cache 1-42
- stateful-session-clustering 1-42
- subscription-durability 1-38, 1-39
- system-id 1-34

T

- table-map 1-44
 - optimistic-column 1-44
 - verify-columns 1-44
- table-name 1-44, 1-45
- taglib 1-33, 1-44
 - taglib-location 1-33

- taglib-uri 1-33
- tlib-version 1-44
- transaction-isolation 1-43
- transaction-type 1-39
 - ejb-class 1-38
 - transaction-type 1-38
- trans-attribute 1-39
- transformer-factory 1-34
- trans-timeout-seconds 1-43
- type-identifier 1-43

U

- uri 1-44
- url-match-map 1-31

V

- validator 1-44
- virtual-directory-mapping 1-31

W

- weblogic-ql 1-46
- weblogic-query
 - group-name 1-46
 - include-updates 1-46
 - max-elements 1-46
 - query-method 1-46
 - sql-select-distinct 1-46
 - weblogic-ql 1-46
- weblogic-rdbms-relation 1-45
- weblogic-relationship-role 1-45
 - group-name 1-45
- welcome-file-list 1-32
- when-to-cache 1-34