



BEA WebLogic Server™

WebLogic Builder Online Help

Release 8.1
Revised: Feb 18, 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 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.

Contents

How WebLogic Builder Works	1-1
Recommended Uses for WebLogic Builder	1-2
Limitations of WebLogic Builder	1-2
Starting WebLogic Builder	1-3
Migrating a J2EE Module to WebLogic Server.	1-6
Working with Web Applications	1-7
Adding a Servlet with Servlet Mapping and Security Constraints	1-7
Adding an ejb-ref/ejb-local-ref	1-9
Adding a Resource Reference.	1-9
Adding a Listener Class	1-10
Adding a Filter with Filter Mapping.	1-10
Defining a Match Map Class	1-11
Setting Welcome and Error Pages.	1-11
Adding an Existing Tag Library	1-12
Adding a Virtual Directory	1-12
Working with EJBs	1-12
Creating a Relationship Between 2.0 CMP Beans	1-13
Adding a CMP Field to an Entity Bean	1-15
Adding a Finder Method to an EJB	1-15
Specifying Optimistic Concurrency	1-16
Adding an ejb-reference or an ejb-local-reference Between Two Beans	1-17
Working with the J2EE Container	1-17

Ordering a Module	1-17
Setting Up EJB Caching	1-18
Choosing a Security Realm	1-18
WebLogic Builder User Interface	1-18
Menu Tasks	1-18
Deployment Descriptor Elements in WebLogic Builder	1-22
Prerequisites	2-2
Procedures	2-2
Step 1: Set up your applications and environment.	2-2
Step 2: Generate deployment descriptors for Smart Ticket.	2-3
Step 3: Specify <context-root> for the web application.	2-4
Step 4: Specify JNDI names.	2-4
Step 5: Define Smart Ticket's data sources.	2-5
Step 6: Deploy Smart Ticket on WebLogic Server.	2-6
Step 7: Run Smart Ticket.	2-7
Big Picture.	2-7
Best Practices	2-7
Related Reading	2-8

WebLogic Builder

This document contains the following sections.

- [“How WebLogic Builder Works” on page 1-1](#)
- [“Recommended Uses for WebLogic Builder” on page 1-2](#)
- [“Limitations of WebLogic Builder” on page 1-2](#)
- [“Starting WebLogic Builder” on page 1-3](#)
- [“Migrating a J2EE Module to WebLogic Server” on page 1-6](#)
- [“Working with Web Applications” on page 1-7](#)
- [“Working with EJBs” on page 1-12](#)
- [“Working with the J2EE Container” on page 1-17](#)
- [“WebLogic Builder User Interface” on page 1-18](#)
- [“Porting and Deploying Applications with WebLogic Builder” on page 2-1](#)

How WebLogic Builder Works

WebLogic Builder is a visual environment for editing a J2EE 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.

Open WebLogic Builder and use the file menu to open an application's compiled J2EE components (*.class files or modules that contain *.class files). If any of the application's deployment descriptor files needed for deployment on WebLogic Server are missing or defective, WebLogic Builder will offer to generate new or repaired descriptor files.

Once deployment descriptor files exist, use WebLogic Builder to edit their elements and attributes. 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

- 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, choose BEA WebLogic Platform-->Development Tools-->WebLogic Builder.

In the command-line, use the following command:

for Windows:

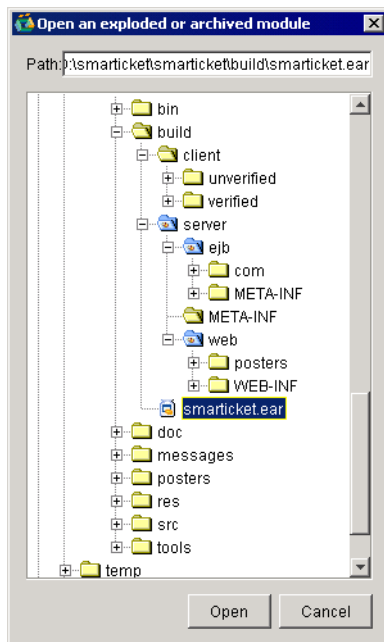
```
startWLBuilder.cmd
```

for Unix:

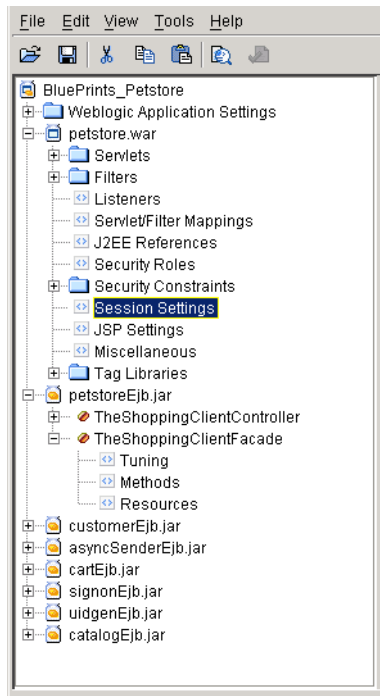
```
startWLBuilder.sh
```

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.

The screenshot shows the 'HTTP Cookie Settings' tab in the WebLogic Builder interface. The dialog box contains the following settings:

- General** | **HTTP Cookie Settings** | **Misc.**
- Idle Timeout (seconds):** 3600
- Session timeout trigger interval (seconds):** 60
- persistentStoreType:** memory
- File Persistence Directory:** session_db
- JDBC Store Pool Name:** (empty field)
- JDBC Timeout (seconds):** 120
- "Cookie" Persistence Name:** WLC COOKIE

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.

To generate the descriptors, WebLogic Builder matches beans with their interfaces using method signatures and naming conventions. WebLogic Builder expects that bean class file names will end in either “Bean” or “EJB.” If possible, Builder will match up both methods and bean names. If bean names do not match, Builder looks for beans and interfaces that contain the same methods. If there are duplicate methods, or if it is not possible to use both method signatures and bean

names, Builder displays a warning in the error pane that it has made a guess based on available information. You can click on this error message to be taken to the class panel of the suspect bean to check and edit the class selection.

WebLogic Builder does not overwrite existing deployment descriptor files.

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

Working with Web Applications

The following sections provide information about Web Applications.

- [Adding a Servlet with Servlet Mapping and Security Constraints](#)
- [Adding an ejb-ref/ejb-local-ref](#)
- [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 an Existing 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

Use the following procedures to define security roles and add security constraints and assignments to them.

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 Roles.
4. In the editing panel, select a Role and click Edit.
5. In the Edit dialog, add the names of members of the Role.
6. Expand the Security Constraints node, and select a Constraint.
7. In the Resources/Pages tab, set the following:
 - Web Resource Name
 - URL patterns for Web Resources
See [virtual-directory-mapping](#) in *weblogic.xml Deployment Descriptor Elements*, and [filter-mapping](#) in *web.xml Deployment Descriptor Elements*.
 - HTTP methods
 - allowed and disallowed roles
8. In the Roles tab, set the Roles for which the Resources/Pages settings are allowed.
9. In the SSL/Misc tab, set:
 - Transport Guarantee
See [security-constraint](#) in *web.xml Deployment Descriptor Elements*.
 - Display Name

Adding an ejb-ref/ejb-local-ref

Use an `ejb-ref` in the web application to define a reference to an EJB resource. See [ejb-ref](#) in *web.xml Deployment Descriptor Elements*.

Use an `ejb-local-ref` to declare a reference to the home interface of a local EJB. See [ejb-local-ref](#) in *web.xml Deployment Descriptor Elements*.

1. In the navigational tree, under the Web Tier, select J2EE References.
2. In the J2EE References editing panel, select the EJB Refs tab for an `ejb-ref`, or the EJB Local Refs tab for an `ejb-local-ref`, and click Add.
3. In the Edit dialog, specify the following properties of the reference and then click OK:
 - Reference Name
 - Link Name (optional)
 - EJB Type (Session or Entity)
 - Remote Interface
 - Home Interface
 - Description (optional)

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

Adding a Resource Reference

Declare a reference to an external resource for your web application using the following procedure. The `resource-ref` element written using this procedure is located in `web.xml`. See [resource-ref](#).

1. In the navigational tree, under the name of your Web Application, select J2EE References.
2. In the J2EE References editing panel, select the Resource Refs tab and click Add.
3. In the Add dialog, select or enter the following parameters for the resource reference:
 - Reference name
 - Reference type
 - JNDI name

- Resource Sharing
- Resource Authentication
- Description

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 Server, and click OK.

Adding a Listener Class

Add an event listener class to a Web Application using the following procedure. See [Application Events and Event Listener Classes](#).

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

Adding a Filter with Filter Mapping

Declare a filter and filter mapping for your Web application using the following procedure. The `filter` element and `filter-mapping` element written using this procedure are located in `web.xml`. See [filter](#) and [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:
 - In the General tab, enter the filter name and class and click OK.
 - In the Init Params tab, add the name, value and, optionally, a description of the parameters, and click OK.
 - In the Display tab, enter the Display Name, Small Icon (must reside within the Web Application), Large Icon (also must reside within the Web .Application), and Description (optional), 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

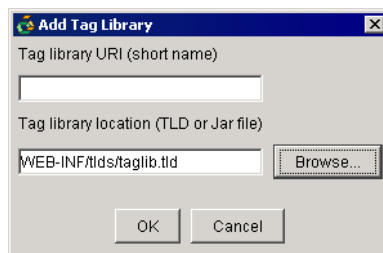
Specify a welcome page and error using the following procedure. The `welcome-file-list` element and `error-page` element written using this procedure are located in `web.xml`. See [welcome-file-list](#) and [error-page](#).

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 an Existing Tag Library

Add tag libraries to your web application using the following procedure. The `taglib` element written using this procedure is located in `web.xml`. For information about tag libraries, see [Overview of Programming JSP Tag Extensions](#). Also see [taglib](#).

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. See [virtual-directory-mapping](#) in *weblogic.xml Deployment Descriptor Elements*.

1. In the navigational tree, under Web Application, Miscellaneous, 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

WebLogic Builder generates descriptors and interfaces for a bean class only if the bean class follows the naming convention of ending in either “Bean” or “EJB.”

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](#)
- [Specifying Optimistic Concurrency](#)
- [Adding an ejb-reference or an ejb-local-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 2.0 beans, WebLogic Builder will not display the Relations node.
2. In the Relations dialog, enter 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, set 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.

Relation 2/3 Role:Role1

* Role name:
Role1

CMR field for RecordingEJB:
bandName

PKField for BandEJB	Columns for RecordingEJB
founder	<specify a column>
name	title

< Back Next > Cancel Help

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

Relation 3/3 Role:Role0

Role name:
Role0

☒ Bidirectional relation

CMR field for BandEJB:
name

CMR field type:

< Back Finish Cancel Help

- 8. Click Finish.

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

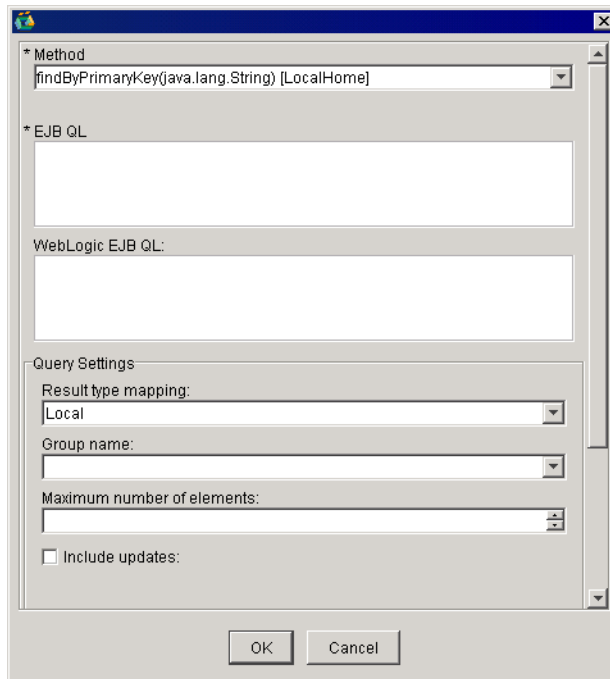
Adding a CMP Field to an Entity Bean

1. In the navigational tree under EJB, expand an entity bean node, select the CMP Fields node and click Add.
2. In the CMP Fields dialog, set the field's name. 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

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. See [Choosing a Concurrency Strategy](#).

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.
2. Click Configure concurrency and select the Verify column and the Optimistic column to map the entity bean to a table.
3. Select Version or Timestamp in the Verify column to enable the Optimistic column field.

4. 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 or an ejb-local-reference Between Two Beans

See http://java.sun.com/dtd/ejb-jar_2_0.dtd and [weblogic-ejb-jar.xml Deployment Descriptor Reference](#).

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

In the Add dialog, enter or select the following parameters:

- EJB name
- EJB reference (or local reference) name
- Type of bean
- Home class
- Remote class
- JNDI name

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. See [Deployment Order](#) in *Deploying WebLogic Server Applications*.

1. In the navigational tree, select the root node of the application.
2. In the 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. See [Tuning WebLogic Server EJBs](#) and <http://www.bea.com/servers/wls810/dtd/weblogic-ejb-jar.dtd>.

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

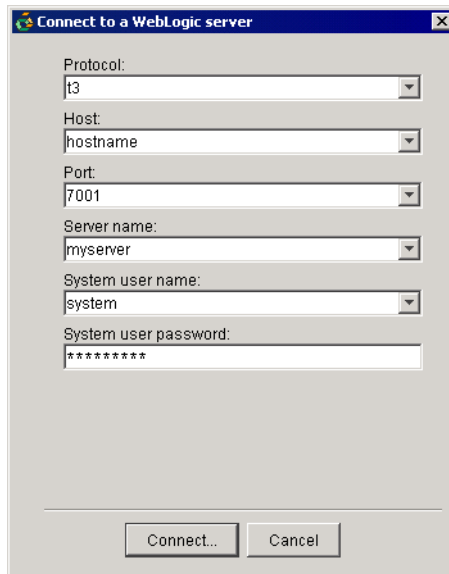
If you try to load a module that is not well formed, for example an EAR that has a nested JAR which is not referenced in the accompanying MANIFEST.MF file, WebLogic Builder may have trouble opening your module.

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.



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.

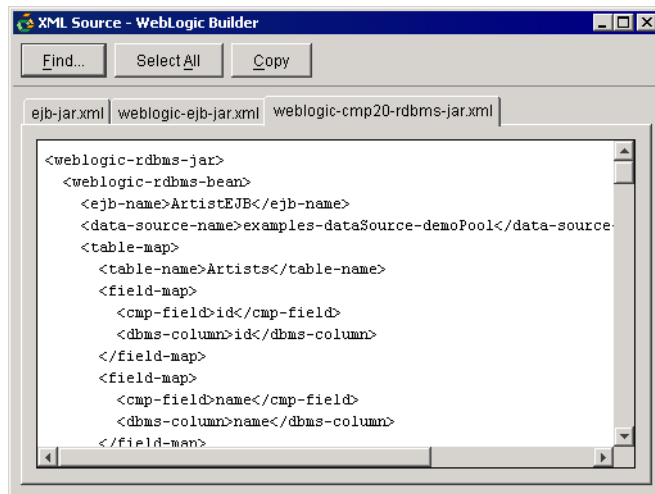
Removing a Deployment Descriptor Element

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.

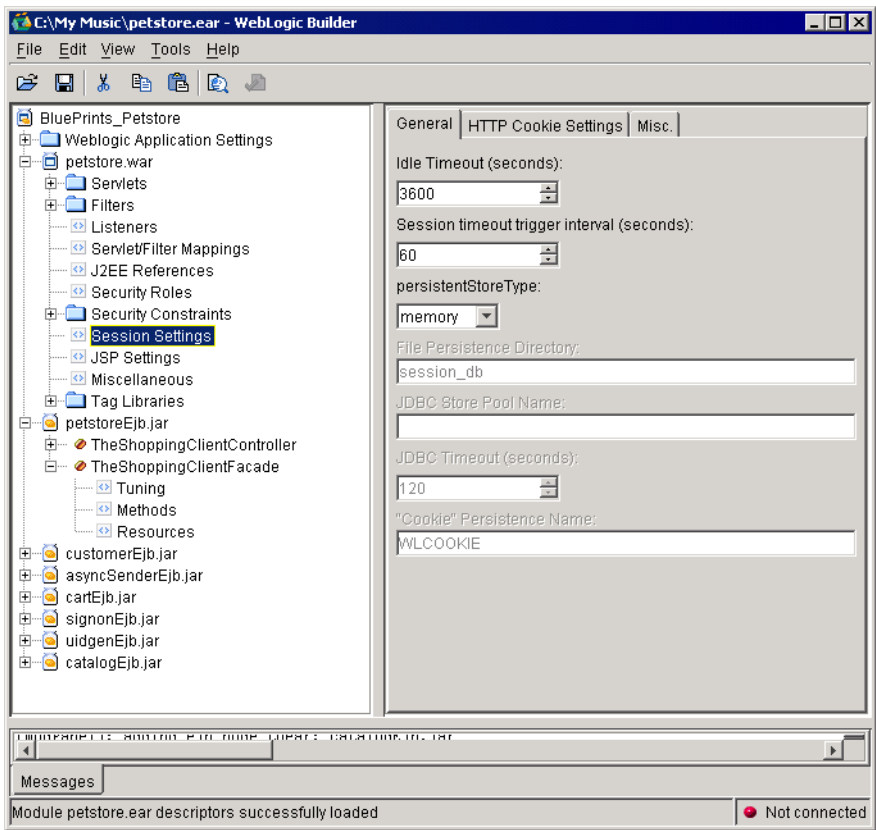


Deployment Descriptor Elements in WebLogic Builder

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
weblogic-version	

XML Element	WebLogic Builder Node —> Tab
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, compilerSupportsEncodi ng, 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

XML Elements and Attributes	WebLogic Builder Node —> Tab
description	Web Application Node—>Display
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-classjsp-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-codeexception-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

XML Elements and Attributes	WebLogic Builder Node —> Tab
env-entry: description, env-entry-name, env-entry-value, env-entry-type	Web Application, J2EE References—>Env Entries
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 Elements and Attributes	WebLogic Builder Node —> Tab
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-able 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
abstract-schema-name	EJB—>Advanced
acknowledge-mode	Message Driven Bean—>Advanced
security-role	EJB—>Security
method-permission,	EJB, Methods—>Permissions
container-transaction	EJB, Methods —>Transactions
cascade-delete	not supported
cmp-field: description, field-name	EJB, CMP Fields—>CMP Fields
cmp-version	EJB —>Persistence
cmr-field: description, cmr-field-name, cmr-field-type	EJB, Relations—>Relation Wizard (right-click Relations)
destination-type	Message Driven Bean—> General
ejb-class	EJB—>Classes
ejb-client-jar	not supported
ejb-link	EJB—>Resources
ejb-local-ref: 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 Deployment Descriptor Reference](#).

XML Elements and Attributes	WebLogic Builder Node —> Tab
cache-between-transactions	EJB, Tuning—>Cache
concurrency-strategy	EJB, Tuning—>Cache
connection-factory-jndi-name	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

XML Elements and Attributes	WebLogic Builder Node —> Tab
max-beans-in-free-pool	EJB, Tuning—>Pool
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:	not supported
home-is-clusterable, home-load-algorithm, home-call-router-class-na me, stateless-bean-is-clustera ble, stateless-bean-load-algori thm, stateless-bean-call-router- class-name, stateless-bean-methods-ar e-idempotent	
stateless-session-descript or: 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 [Programming WebLogic Server Enterprise JavaBeans](#).

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
data-source-name	EJB—>Persistence
table-name	EJB Application, Relations—>Relation wizard (right-click Relations)

XML Elements and Attributes	WebLogic Builder Node —> Tab
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-role, 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)
relationship-caching: caching-name, caching-element	not supported
caching-name	not supported

XML Elements and Attributes	WebLogic Builder Node —> Tab
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
delay-database-insert-until	EJB—>Advanced
validate-db-schema-with	not supported
database-type	not supported

Porting and Deploying Applications with WebLogic Builder

This tutorial shows you how to rapidly deploy Sun's BluePrint wireless application Smart Ticket to WebLogic Server using the WebLogic Builder utility. WebLogic Builder is a visual environment that enables you to generate and edit deployment descriptor files. Smart Ticket is a J2EE application that lets you browse and purchase movie tickets using a wireless device such as a cell phone. Smart Ticket has both web application and EJB components.

In particular, the tutorial shows how to:

- Generate and edit Smart Ticket's deployment descriptor files using WebLogic Builder
- Configure Smart Ticket's data sources using the WebLogic Server Administration Console
- Run Smart Ticket on WebLogic Server using a device emulator in the J2ME Wireless Toolkit

This tutorial contains the following sections:

- [“Prerequisites” on page 2-2](#)
- [“Procedures” on page 2-2](#)
- [“Big Picture” on page 2-7](#)
- [“Best Practices” on page 2-7](#)
- [“Related Reading” on page 2-8](#)

Prerequisites

Before starting this tutorial:

- Read about Java Smart Ticket 1.1.1 at <http://java.sun.com/blueprints/wireless/>.
- Read about WebLogic Builder at [WebLogic Builder Online Help](#).
- Install Java 2 Platform, Micro Edition 1.0.4_01 at <http://java.sun.com/j2me/docs/>.
- Install and start WebLogic Server. See <http://www.bea.com>.

Procedures

Follow these steps to deploy Smart Ticket on WebLogic Server.

- [Step 1: Set up your applications and environment.](#)
- [Step 2: Generate deployment descriptors for Smart Ticket.](#)
- [Step 3: Specify <context-root> for the web application.](#)
- [Step 4: Specify JNDI names.](#)
- [Step 5: Define Smart Ticket's data sources.](#)
- [Step 6: Deploy Smart Ticket on WebLogic Server.](#)
- [Step 7: Run Smart Ticket.](#)

Step 1: Set up your applications and environment.

In this step you install Smart Ticket and J2ME, Sun's Wireless Toolkit. You also set up your environment so you can build the application.

1. If you have not already done so, install and start WebLogic Server 8.1.

Download WebLogic Server 8.1 from <http://www.bea.com> and install it. The WebLogic Server installation directory, `c:\bea\weblogic810` unless you specify another location, is called `WL_HOME`.

2. Download and install Smart Ticket 1.1.1.

Download the Smart Ticket demo application source code at <http://developer.java.sun.com/developer/releases/smartticket/>. Extract it into a new directory

on the same machine or accessible from `WL_HOME`. We will call this directory `SMARTICKET_HOME`.

3. Download and install J2ME to a location we will call `J2MEWTK_HOME` (say, `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/jdk141` 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\domains\examples`.
6. Open `SMARTICKET_HOME\smarticket\localant.bat` and add `"%CLASSPATH%"` to the end of the `ANT_CLASSPATH` line.
7. In the `SMARTICKET_HOME\smarticket` directory, build Smart Ticket by running `localant.bat`. WebLogic Builder requires compiled `.class` files and cannot use `.java` files.

Step 2: Generate deployment descriptors for Smart Ticket.

In this sequence, WebLogic Builder reads the deployment descriptor files that are included with the Smart Ticket application and introspects the application's `.class` files to create the deployment descriptor files that help an application to run on WebLogic Server.

WebLogic Builder will not overwrite the application's existing deployment descriptor files.

1. Open WebLogic Builder. In Windows select Start->Programs->BEA WebLogic Platform->WebLogic Server 8.1 | WebLogic Builder. In UNIX use the following command:
`startBuilder.sh`
2. In WebLogic Builder's File->Open menu, select the folder `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 to have WebLogic Builder introspect the Smart Ticket class files and generate `weblogic.xml` and `weblogic-ejb-jar.xml`.
3. Select File->Save.

4. In WebLogic Builder, create `smarticket.ear` by selecting File->Archive and specifying `SMARTICKET_HOME\bin\smarticket.ear`.

`smarticket.ear` is the application archive that you will configure and then deploy on WebLogic Server.

Step 3: Specify `<context-root>` for the web application.

In this step, you set the Smart Ticket web application's context path using the `<context-root>` element.

1. In WebLogic Builder, select the `\web` node's Context Path tab.
2. In the Context Path text field, enter `SmarTicketApp`.

Now the application's `<context-root>` element is specified.

Step 4: Specify JNDI names.

In this sequence, you assign JNDI names to references in the web application and in EJBs so that the data in the entity beans is accessible from the web application.

- to the web application's EJB references and resource references, and
- to the EJB resource references

1. Select Builder's EJB Refs panel in the J2EE Refs node, and specify the JNDI names for the EJB Refs as follows:

Reference Name	EJB Type	JNDI Name
<code>ejb/MovieInfo</code>	<code>Session</code>	<code>MovieInfo</code>
<code>ejb/TicketSales</code>	<code>Entity</code>	<code>TicketSales</code>
<code>ejb/Customer</code>	<code>Entity</code>	<code>Customer</code>
<code>ejb/LocaleInfo</code>	<code>Session</code>	<code>LocaleInfo</code>

2. In the Resource Refs panel in Builder's J2EE Refs node, set the Ref Name, EJB Type, and JNDI Name according to the following table. Set the Resource Authentication Type to Container.

Reference Name	Reference Type	JNDI Name
jdbc/MovieInfoDataSou rce	java.sql.DataSource	MovieInfoDataSource
jdbc/TicketSalesDataS ource	java.sql.DataSource	TicketSalesDataSource
jdbc/CustomerDataSou rce	java.sql.DataSource	CustomerDataSource
jdbc/LocaleInfoDataSo urce	java.sql.DataSource	LocaleInfoDataSource

3. In the Resource References panel from Builder's EJB Resources node, set JNDI names for EJB Resources according to the following table.

Resource Reference Name	Resource Reference Type	JNDI Name
jdbc/MovieInfoDataSou rce	java.sql.DataSource	MovieInfoDataSource
jdbc/TicketSalesDataS ource	java.sql.DataSource	TicketSalesDataSource
jdbc/CustomerDataSou rce	java.sql.DataSource	CustomerDataSource
jdbc/LocaleInfoDataSo urce	java.sql.DataSource	LocaleInfoDataSource

4. In Builder, select File->Save to save changes to the archive.

Step 5: Define Smart Ticket's data sources.

In this step, start the Examples server and use the WebLogic Server Administration Console to configure the Data Sources for each of the four EJBs the Smart Ticket application uses. Also in this step, replace Smart Ticket's Cloudscape database with the Pointbase RDBMS included with WebLogic Server.

The JNDI names for the data sources must match the JNDI names for the web application resource references and the EJB resources you set in Step 4.

1. Start the Examples server from: Start->Programs->BEA WebLogic Platform-> WebLogic Server 8.1->Server Tour and Examples->Launch Examples Server.

The Examples Server launches the WebLogic Server Examples page.

2. Open the WebLogic Server 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`.
3. Select the JDBC node and click Tx Data Sources.
4. Select Configure a new JDBC Tx Data Source.
5. 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 Server examples use. Click Create.
6. Click the Targets Tab, then select the `examplesServer` in the Available column and click on the right arrow to target it. Click Apply.
7. Repeat steps 4 and 5 for the other three data sources (`MyMovieInfoDataSource`, `MyLocaleInfoDataSource`, and `MyTicketSalesDataSource`).
8. To replace Smart Ticket's Cloudscape database with Pointbase, the evaluation RDBMS included with WebLogic Server 8.1, 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=%POINTBASEHOME%  
com.pointbase.jdbc.jdbcUniversalDriver -u examples -p examples -verbose  
./src/smarticketPointBase.sql
```
9. We are also making the `smarticket.sql` Pointbase friendly by replacing it with a script that substitutes "int" data types with "integer." Extract `smarticketPointBase.sql` from <http://edocs.bea.com/wls/docs81/quickstart/smarticket-patch.zip> and copy it into `SMARTICKET_HOME\smarticket\src`.
10. Set up the database by running `SMARTICKET_HOME\smarticket\populate.bat`.

Step 6: Deploy Smart Ticket on WebLogic Server.

Now you are ready to deploy and run Smart Ticket on the WebLogic Server 8.1 Examples Server.

1. Specify the server for the application using the Connect to Server dialog in WebLogic Builder's Tools menu.
2. Deploy Smart Ticket by selecting Deploy Module from the WebLogic Builder tools menu.
3. Set the port in Smart Ticket's executable, SMARTTICKET_HOME\bin\smartticket.jad, by replacing the given port number (8000) with the port number of the Examples server, which by default is 7001.
4. Optionally, open the J2ME Wireless Toolkit and select the default device and specify preferences. There are several options available for using the emulator.
5. Start SMARTTICKET_HOME\bin\smartticket.jad by double-clicking it or by selecting Start->J2ME Wireless Application->Run MIDP Application and selecting it.

A J2ME wireless emulator, either the one you selected in step 4 or the default emulator, DefaultGrayPhone, appears. The Smart Ticket asks you to create a user account and log in.

Step 7: Run Smart Ticket.

When creating a user account for the Smart Ticket Application, enter 95130 or 95054 for your zip code. Your password must be six characters long.

Try selecting the 'Poster' mode when you create an account; this will enable your phone or other wireless emulator to display a preview poster for the movies as you browse them.

Big Picture

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](http://edocs.bea.com/wls/docs81/wlbuilder/index.html) at <http://edocs.bea.com/wls/docs81/wlbuilder/index.html>.

The sample application Smart Ticket illustrates how the J2EE platform interoperates with the J2ME platform to create enterprise applications that serve mobile client devices, such as cell phones, two-way pagers, and palmtops.

Best Practices

View the progress and status of the Smart Ticket application on WebLogic Server by monitoring the Launch Examples Server command window that appeared when you started the server.

When selecting a device emulator from the J2ME Wireless Toolkit's Default Device menu, the DefaultGreyPhone is the easiest to work with.

If you are redeploying the Smart Ticket application, clear the database of the previous user's information. To do so:

1. Run the J2ME Wireless Ticket Utility (available from the Windows start menu, or as an executable in `J2ME_HOME/bin`).
2. Click Clean Database.
3. Restart Smart Ticket by double-clicking `SMARTTICKET_HOME\bin\smartticket.jad` or from Start->J2ME Wireless Application->Run MIDP Application.

Related Reading

See the procedure for porting Smart Ticket to WebLogic Server 8.1 without using WebLogic Builder: [Java Smart Ticket Demo 1.1](http://edocs.bea.com/wls/docs81/quickstart/smartticket.html) at <http://edocs.bea.com/wls/docs81/quickstart/smartticket.html>.

Index

A

- abstract-schema-name 1-31
- acknowledge-mode 1-31
- ANT_CLASSPATH 2-3
- automatic-key-generation 1-39, 1-41
 - generator-name 1-41
 - generator-type 1-41
 - key-cache-size 1-41

C

- cache-between-transactions 1-34
- cache-timeout-interval 1-29
- caching-element 1-41
- caching-name 1-40
- caching-strategy 1-28
- cascade-delete 1-31
- charset-params 1-25
- check-exists-on-method 1-39
- clients 1-36
- clients-on-same-server 1-36
- cmp-field 1-31, 1-40
 - description 1-31
 - field-name 1-31
- cmp-version 1-31
- cmr-field 1-31
 - cmr-field-name 1-31
 - cmr-field-type 1-31
 - description 1-31
- column-map 1-40
- concurrency-strategy 1-34
- connection-factory 1-30
- connection-factory-jndi-name 1-34

- container-descriptor 1-25
- container-transaction 1-31
- context-param 1-26
- context-root 2-4
- create-default-dbms-tables 1-39

D

- dbms-column 1-40
- dbms-column-type 1-40
- delay-database-insert-until 1-39
- delay-updates-until-end-of-tx 1-35
- demoPool 2-6
- deployment descriptor files, WebLogic Builder
 - will not overwrite existing 1-7
- description 1-26, 1-39
- destination-jndi-name 1-35
- destination-type 1-31
- display-name 1-25, 1-39
- distributable 1-26
- document-builder-factory 1-29

E

- EJB Refs 2-4
- EJB Resources 2-5
- ejb-class 1-31, 1-32
 - home 1-32
 - local 1-32
 - local-home 1-32
 - remote 1-32
- ejb-client-jar 1-31
- ejb-link 1-28, 1-31
- ejb-local-ref 1-31

- description 1-31
- ejb-link 1-31
- ejb-ref-name 1-31
- ejb-ref-type 1-31
- local 1-31
- local-home 1-31
- ejb-local-reference-description 1-35
- ejb-name 1-32, 1-39
- ejb-ql 1-32
- ejb-ref 1-28, 1-32
 - description 1-32
 - ejb-link 1-32
 - home 1-32
 - remote 1-32
- ejb-reference-description 1-35
 - ejb-ref-name 1-35
 - jndi-name 1-35
- ejb-ref-name 1-28
- ejb-ref-type 1-28
- ejb-relation 1-32
 - description 1-32
 - ejb-relation-name 1-32
 - ejb-relationship-role 1-32
- ejb-relationship-role 1-32
 - cmr-field 1-32
 - ejb-relationship-role-name 1-32
 - relationship-role-source 1-32
- ejb-relationship-role-name 1-32
- enable-call-by-reference 1-35
- enable-dynamic-queries 1-35, 1-36
- entity-cache 1-28, 1-35
 - cache-between-transactions 1-35
 - concurrency-strategy 1-35
 - idle-timeout-seconds 1-35
 - max-beans-in-cache 1-35
 - read-timeout-seconds 1-35
- entity-cache-name 1-28
- entity-clustering 1-36
 - home-call-router-class-name 1-36
 - home-is-clusterable 1-36

- home-load-algorithm 1-36
- entity-mapping 1-29
- entity-mapping-name 1-29
- entity-uri 1-29
- env-entry 1-28
- env-entry-name 1-28
- env-entry-type 1-28
- env-entry-value 1-28
- error-page
 - error-codeexception-type 1-26
 - location 1-26
- examplesServer 2-6

F

- field-group 1-39
- field-map 1-39, 1-40
 - cmp-field 1-40
 - dbms-column 1-40
 - dbms-column-type 1-40
- filter 1-26
 - display-name 1-26
 - filter-name 1-26
 - icon 1-26
- filter-class 1-26
- filter-mapping 1-26
- finders-load-bean 1-36
- foreign-key-column 1-40
- foreign-key-table 1-40

G

- generator-name 1-41
- generator-type 1-41

H

- home 1-34
- home-call-router-class-name 1-36, 1-37
- home-is-clusterable 1-36, 1-37
- home-load-algorithm 1-36, 1-37

I

- icon 1-25
- idempotent-methods 1-36
- idle-timeout-seconds 1-36, 1-37
- include-updates 1-41
- initial-beans-in-free-pool 1-36, 1-37
- initial-context-factory 1-36
- init-param 1-26
- invalidation-target 1-38
 - ejb-name 1-38
- isolation-level 1-36, 1-38

J

- J2EE Refs 2-4
- J2MEWTK_HOME 2-3
- jdbc-connection-pool 1-30
 - acl-name 1-30
 - connection-factory
 - connection-properties 1-30
 - factory-name 1-30
 - data-source-name 1-30
 - driver-params 1-30
 - prepared-statement 1-30
 - row-prefetch-enabled 1-30
 - row-prefetch-size 1-30
 - statement 1-30
 - stream-chunk-size 1-30
 - pool-params 1-30
 - connection-check-params 1-30
 - leak-profiling-enabled 1-30
 - login-delay-seconds 1-30
 - size-params 1-30
 - xa-params 1-30
 - xa-params 1-30
 - debug-level 1-30
 - end-only-once-enabled 1-30
 - keep-conn-until-tx-complete-enabled 1-30
 - keep-logical-conn-open-on-release 1-30

- local-transaction-supported 1-30
- new-conn-for-commit-enabled 1-30
- prepared-statement-cache-size 1-30
- recover-only-once-enabled 1-30
- resource-health-monitoring-enabled 1-30
- tx-context-on-close-needed 1-30

- jms-client-id 1-35
- jms-polling-interval-seconds 1-35
- jndi-name 1-36
- jsp-descriptor 1-25
 - jsp-param 1-25
 - param-name 1-25
 - compileCommand 1-25
 - compileFlags 1-25
 - compilerClass 1-25
 - compilerSupportsEncoding 1-25
 - debug 1-25
 - defaultFilename 1-25
 - keepgenerated 1-25
 - noTryBlocks 1-25
 - packagePrefix 1-25
 - pageCheckSeconds 1-25
 - precompile 1-25
 - verbose 1-25
 - workingDir 1-25
- jsp-file 1-26
- jsp-version 1-39

K

- key-cache-size 1-41
- key-column 1-40

L

- large-icon 1-39
- listener 1-26, 1-39
- load-on-startup 1-26
- localant.bat 2-3
- local-jndi-name 1-36
- login-config 1-27

- auth-method 1-27
- form-login-config 1-27
- realm-name 1-27

M

- max-beans-in-cache 1-28, 1-36, 1-37
- max-beans-in-free-pool 1-37
- max-cache-size 1-28
- max-elements 1-41
- message-driven 1-33
 - ejb-name 1-33
- message-driven-descriptor 1-37
- message-driven-destination 1-33
- message-selector 1-33
 - acknowledge-mode 1-33
- method-permission, 1-31
- mime-mapping 1-26
 - extension 1-26
 - mime-type 1-26
- multiplicity 1-32

O

- optimistic-column 1-40

P

- parser-factory 1-29
- persistence-type 1-33
- persistence-use 1-37
- Pointbase 2-6
- pool 1-37
- populate.bat 2-6
- primary-key-table 1-40
- prim-key-class 1-33
- primkey-field 1-32, 1-33
- principal-name 1-37
- provider-url 1-38
- public-id 1-29

Q

- query 1-33
 - description 1-33
 - ejb-ql 1-33
 - result-type-mapping 1-33
- query-method 1-33, 1-41

R

- read-timeout-seconds 1-37
- realm-name 1-30
- reentrant 1-33
- reference-descriptor 1-24
 - ejb-reference-description 1-24
 - ejb-ref-name 1-24
 - jndi-name 1-24
 - res-env-ref-name 1-24
 - resource-description 1-24
 - resource-env-description 1-24
 - res-ref-name 1-24
- relation-name 1-40
- relationship-caching 1-40
- relationship-role-map 1-40
- relationship-role-name 1-40
- relationships 1-33
 - description 1-33
 - ejb-relation 1-33
- replication-type 1-37
- Resource Authentication Type 2-4
- Resource References 2-5
- Resource Refs 2-4
- resource-env-ref 1-27, 1-32, 1-33
 - description 1-33
 - env-entry 1-32
 - env-entry-name 1-32
 - env-entry-type 1-32
 - env-entry-value 1-32
 - resource-env-ref-name 1-27, 1-33
 - resource-env-ref-type 1-27, 1-33
- resource-ref 1-27, 1-33
 - description 1-33

- res-auth 1-27, 1-33
- res-ref-name 1-27, 1-33
- res-sharing-scope 1-27, 1-33
- res-type 1-27, 1-33
- role-name 1-33, 1-37
- run-as 1-28

S

- saxparser-factory 1-29
- security-constraint 1-27
 - auth-constraint 1-27
 - display-name 1-27
 - user-data-constraint 1-27
 - web-resource-collection 1-27
- security-permission 1-25
- security-role 1-27, 1-31
 - description 1-27
 - role-name 1-27
- security-role-assignment 1-24, 1-37
 - principal-name 1-24
 - role-name 1-24
- security-role-ref 1-26, 1-27
 - description 1-27
 - role-link 1-27
 - role-name 1-27
- servlet 1-26
- servlet-class 1-26
- servlet-mapping 1-26
 - servlet-name 1-26
 - url-pattern 1-26
- servlet-name 1-26
- session 1-34
 - ejb-class 1-34
 - ejb-local-ref 1-34
 - ejb-name 1-34
 - ejb-ref 1-34
 - env-entry 1-34
 - local 1-34
 - local-home 1-34
 - remote 1-34
 - resource-env-ref 1-34
 - resource-ref 1-34
 - security-identity 1-34
 - security-role-ref 1-34
 - session-type 1-34
 - transaction-type 1-34
- session-config 1-26
- session-descriptor 1-24
 - param-name 1-24
 - CacheSize 1-24
 - ConsoleMainAttribute 1-24
 - CookieComment 1-24
 - CookieDomain 1-24
 - CookieMaxAgeSecs 1-24
 - CookieName 1-24
 - CookiePath 1-24
 - CookiesEnabled 1-24
 - IDLength 1-24
 - InvalidationIntervalSecs 1-24
 - JDBCCConnectionTimeoutSecs 1-24
 - PersistentStoreCookieName 1-24
 - PersistentStoreDir 1-24
 - PersistentStorePool 1-24
 - PersistentStoreType 1-24
 - SwapIntervalSecs 1-24
 - TimeoutSecs 1-24
 - TrackingEnabled 1-24
 - URLRewritingEnabled 1-24
 - session-param 1-24
- session-timeout 1-26
- session-type 1-34
- short-name 1-39
- small-icon 1-39
- Smart Ticket 1.1 2-2
- smartticket.ear 2-4
- smartticket.jad 2-7
- smartticket.sql 2-6
- SMARTTICKET_HOME 2-3
- smartticketPointBase.sql 2-6
- sql-select-distinct 1-41
- start-mdbs-with-application 1-28

- stateful-session-cache 1-37
- stateful-session-clustering 1-37
- subscription-durability 1-33, 1-34
- system-id 1-29

T

- table-map 1-39
 - optimistic-column 1-39
 - verify-columns 1-39
- table-name 1-39, 1-40
- taglib 1-27, 1-39
 - taglib-location 1-27
 - taglib-uri 1-27
- tlib-version 1-39
- transaction-isolation 1-38
- transaction-type 1-34
 - ejb-class 1-33
 - transaction-type 1-33
- trans-attribute 1-34
- transformer-factory 1-29
- trans-timeout-seconds 1-38
- Tx Data Sources 2-6
- type-identifier 1-38

U

- uri 1-39
- url-match-map 1-25

V

- validator 1-39
- virtual-directory-mapping 1-25

W

- WebLogic Server Administration Console 2-5, 2-6
- weblogic.xml 2-3
- weblogic-ejb-jar.xml 2-3
- weblogic-ql 1-41

- weblogic-query
 - group-name 1-41
 - include-updates 1-41
 - max-elements 1-41
 - query-method 1-41
 - sql-select-distinct 1-41
 - weblogic-ql 1-41
- weblogic-rdbms-relation 1-40
- weblogic-relationship-role 1-40
 - group-name 1-40
- welcome-file-list 1-26
- when-to-cache 1-29