BEA WebLogic Workshop™ Help

Version 8.1 SP4
December 2004

Copyright

Restricted Rights Legend

Trademarks or Service Marks

Table of Contents

Deploying an Application to a Production Server

The following topic explains the basic concepts of deploying WebLogic Workshop applications to WebLogic Server running in production mode.

For step−by−step instructions on deploying a WebLogic Workshop application to a production server see How Do I: Deploy a WebLogic Workshop Application to a Production Server?.

Development Mode and Production Mode

When you are developing, deploying and testing an application with WebLogic Workshop, the instance of WebLogic Server you are deploying to runs by default in *development mode.* In development mode, WebLogic Server behaves in ways that make it easier to iteratively develop and test an application: it automatically deploys the current application in an exploded format, server resources such as databases and JMS queues necessary for the application to run are automatically created, etc.

When the development cycle is complete, and the application is ready for use, you deploy it to an instance (or instances) of WebLogic Server running in *production* mode. In production mode applications are not automatically deployed and the server resources necessary for running an application are not automatically generated.

For detailed information on starting WebLogic Server in production or development mode see startWebLogic Command.

EAR Files

WebLogic Workshop produces J2EE enterprise applications for deployment to a production server. In an archived format, these are either EAR files if you are deploying an entire application; or, if you are deploying a specific project within an application, a JAR file (provided that specific project is a custom Java control or a Schema project). Note that you cannot deploy a web application project alone; you can deploy a web application project only as part of an entire application.

You can generate an EAR file for a WebLogic Workshop application either (1) from the menu bar by selecting ***Build−−>Build EAR*** or (2) by using the ***wlwBuild.cmd*** command line tool. The wlwBuild.cmd line tool is somewhat more flexible in that you can set flags to build a JAR file for a specific project, instead of building an EAR file for the entire application.

For information generating an ANT build.xml file that calls wlwBuild.cmd, see How Do I: Call wlwBuild.cmd from an ANT build.xml file?

EAR files can deployed to WebLogic Server using either (1) the WebLogic Server console, or (2) the weblogic.Deployer utility.

To use the WebLogic Server console to deploy an EAR file, start the console, expand the Deployments node in the left−hand pane, right−click the Applications node, and select Deploy a new Application.

To use the weblogic.Deployer utility see the Deployment Tools Reference in the WebLogic Server 8.1 documentation.

When you compile an EAR file using Build EAR, a wlw−manifest.xml file is produced and placed in the application's META−INF directory. This wlw−manifiest.xml file lists the server resources that must be created on the production server for the application EAR to run successfully. See the next section for information relating to the wlw−manifest.xml file.

Note: Values specified in a project's WEB−INF/wlw−config.xml file, such as hostname, http−port, and https−port, will be hard−coded into the EAR file. The result will be an EAR file that can be run only on the machine named in the wlw−config.xml file. For this reason, it is recommended that you do not write to the wlw−config.xml file before producing an EAR file. If you need to override the hostname and ports dynamically determined by the server at runtime, use the wlw−runtime−config.xml file instead of wlw−config.xml.

Manual Creation of Server Resources

When deploying EAR files to a production server, a certain amount of manual resource creation is necessary. When an application is built in an EAR file, a wlw−manifest.xml file is produced and placed in the application's META−INF directory. This file lists the JMS queues and database tables that need to be manually created on the target WebLogic Server for the application to run properly.

Note: When you are developing and testing an application with WebLogic Workshop, the creation of the necessary JMS queues and datatables on WebLogic Server takes places automatically on demand.

Required database tables are indicated by a ***<con:conversation−state−table />*** tag. These tables are used by web services to store conversational state. For each occurrence of the <con:conversation−state−table /> tag in the wlw−manifest.xml file, you must create a corresponding datatable on WebLogic Server. For detailed information about the schema required for these tables, see How Do I: Deploy a WebLogic Workshop Application to a Production Server?

Required JMS queues are indicated by pairs of ***<con:async−request−queue>*** and ***<con:async−request−error−queue>*** tags. For each occurrence of these tags in the wlw−manifest.xml file, you must create a corresponding JMS queue on WebLogic Server *and* you must associate the members of the pair by referencing the <con:async−request−error−queue> in the ErrorDestination attribute of the <con:async−request−queue>. For detailed information about how to create these queues, see How Do I: Deploy a WebLogic Workshop Application to a Production Server?

Optionally, you may want to enforce role restrictions on any controls that receive external callbacks. Controls that can receive external callbacks are indicated within a ***<con:external−callbacks/>*** tag in the wlw−manifest.xml file. Since the compilation process turns control files into individual methods on an EJB, you enforce the role restrictions on these post−compilation EJB methods. For detailed information about applying role restrictions in this case see How Do I: Deploy a WebLogic Workshop Application to a Production Server?

Related Topics

Deployment Tools Reference

How Do I: Deploy a WebLogic Workshop Application to a Production Server?

Deployment and Clustering

WebLogic Workshop applications are deployed to WebLogic Server in the form of enterprise application archive (EAR) files. You can deploy a WebLogic Workshop EAR either to a single instance of WebLogic Server or to a cluster of WebLogic Servers.

Cluster configurations do not support iterative development. Therefore, if you want to develop and test applications against a WebLogic server, configure WebLogic Workshop to use a single server belonging to a workshop enabled domain running in development mode. For more information, see How Do I: Create a New WebLogic Workshop−Enabled WebLogic Server Domain?

The topics in this section explain how to generate an EAR file for a WebLogic Workshop project and deploy it, either to a single instance of WebLogic Server, or to a cluster of WebLogic Servers.

Topics Included in this Section

Deploying an Application to a Production Server

Explains the basic concepts of deploying a WebLogic Workshop application to WebLogic Server.

How Do I: Deploy a WebLogic Workshop Application to a Production Server?

Gives step by step instructions for deploying a Workshop application to a production server.

How Do I: Call wlwBuild.cmd from an ANT build.xml file?

Gives step by step instructions for generating an ANT file that compiles a WebLogic Workshop application.

Overview: Clustering

Explains the basic concepts of clustering WebLogic Servers.

Clustering Workshop Applications

Explains the basic concepts of deploying WebLogic Workshop applications to a cluster of WebLogic Servers.

Related Topics

How Do I: Configure a Cluster for a WebLogic Workshop Application?

How Do I: Deploy a WebLogic Workshop Application to a Production Server?

Overview: Clustering

A WebLogic Server cluster is a deployment in which multiple copies, or *instances*, of an application work together to provide increased performance, especially in high traffic contexts. In cases where an application receives a high volume of requests, the different instances of WebLogic Server in the cluster share the work of processing the requests. From the client's point of view, there appears to be only one instance of WebLogic Server servicing the requests.

Clusters also provide failover support. Should one instance of the application fail for some reason−−for example, because of a hardware outage−−another copy of the application in the cluster can pick up and complete the tasks left incomplete by the failed server.

The server instances that make up the cluster can run on a single machine, or they can run on different machines. Each server instance in a cluster must run the same version of WebLogic Server.

To learn more about deploying applications to WebLogic Server clusters see WebLogic Server Clusters in the WebLogic Server 8.1 Documentation.

Related Topics

Clustering Workshop Web Services

Clustering Workshop Applications

Clusters provide scalability and support failover for web resources. The basic clustering model consists of the following elements:

1. One administration server that manages state and configures the other servers in the cluster
2. One HTTP proxy server—either a hardware or a software proxy server—which receives requests from clients and distributes jobs to the other servers in the cluster
3. Any number of managed servers that actually do the work of servicing requests from clients

All configuration of the cluster takes place on the administration server: all other servers in the cluster use the copy of config.xml on the administration server. (There may be local copies of config.xml on the managed servers in the cluster, but, these copies of config.xml are ignored in favor of the copy on the administration server.)

The following three required WebLogic Workshop resources must also be deployed homogeneously across all servers in a cluster:

- JDBCConnectionPool
- JDBCTxDatasource
- JMSQueueConnectionFactory.

A JMS Server is also a required Workshop resource, however, it can only be deployed to one server in the cluster.

Configuring Clusters in config.xml

Complete syntax for the config.xml file can be found at WebLogic Server Configuration Reference in the WebLogic Server 8.1 documentation.

The following sections highlight some of the most important elements within a cluster−defining config.xml file (located on the administration server), including the <Cluster> element, resource deployment, and database support.

The <Cluster> Element

The ClusterAddress attribute specifies a DNS name that maps to the list of IPs of the servers. ClusterAddress does not give the DNS name of the multicast address, which does not require a DNS name. The cluster as a whole can be used as a deployment target. To deploy a J2EE resource to the entire cluster, use the value of the Name attribute as the target of the deployment. To learn more, see the −targets parameter of the deployment tool, Deployment Tools Reference, in the WebLogic Server 8.1 documentation.

Resource Deployment

Resources in the cluster—such as database connection pools, data sources, and JMS servers—are defined in the administration server's config.xml file. Resources are not by default universally available across the cluster. The servers they are deployed to are specified by their Targets attributes.

The connection pool, conversational datasource, and queue connection factory that Workshop relies on is defined on each managed server in the cluster. For example, the Targets attribute on the JDBConnectionPool and JDBCDataSource elements specifies each managed server, and each managed server has its own pool of connections.

However, the JMSServers can only be targeted at one server in the cluster. Currently Workshop only uses one JMSServer that is targeted, by convention, at the first managed server in the cluster.

Proxy Server Setup

Clusters can use a software proxy server to distribute HTTP requests across the cluster. The proxy server is also called the *sprayer* or *load balancer*. This software proxy is implemented as a web application deployed to the proxy server. You configure the proxy by editing the web.xml descriptor in the proxy application's WAR file. There are entries in the descriptor for specifying what IP addresses and ports the proxy should distribute requests to. For more information about configuring the proxy server see Configure Proxy Plug−Ins in the WebLogic Server 8.1 documentation.

When using a proxy, it is necessary to set the hostname, HTTP, and HTTPS ports on the target cluster, otherwise the target cluster will not know how to interpret the requested URLs coming from the proxy. You set the hostname and ports by (1) setting the FrontEnd information through the WebLogic Server console and (2) in the target cluster's the wlw−runtime−config.xml file.

Editing the application's wlw−config.xml file is not recommended, because these values are fixed at compile−time. It is generally best to configure the hostname and ports through the wlw−runtime−config.xml file, which overrides the values in the wlw−config.xml.

To set the FrontEnd host and port information using the WebLogic Server console, open the console, and navigate to

[your_domain]−−>Servers−−>[your_server]−−>Protocols tab−−>HTTP tab−−>Advanced Options

Then edit the Frontend Host, Frontend HTTP port, and Frontend HTTPS port fields. Note that the Frontend host must be set on each managed server in the cluster, but should not be set on the administration server. All servers in the cluster must be restarted for this change to take effect. Also see Configuring WebLogic Server Web Components in the WebLogic Server 8.1 documentation.

To set the host and port information in the wlw−runtime−config.xml file see wlw−runtime−config.xml in the WebLogic Workshop reference documentation.

Related Topics

How Do I: Configure a Cluster for a WebLogic Workshop Application?

Deploying Portal Applications

There are many considerations and best practices involved in moving a portal application from the development to testing to production. For detailed information and guidance on the deployment process, see "Deploying Portal Applications" on e−docs at http://e−docs.bea.com/wlp/docs81/deploy/index.html.

Related Topics

Deploying Applications

Deployment and Clustering