The Web Applications Container

A Web application contains an application’s resources, such as servlets, JavaServer Pages (JSPs), JSP tag libraries, and any static resources such as HTML pages and image files. A Web application adds service-refs (Web services) and message-destination-refs (JMS destinations/queues) to an application. It can also define links to outside resources such as Enterprise JavaBeans (EJBs).

**Web Applications and Java EE**

An important aspect of the [Java Platform, Enterprise Edition (Java EE) Version 5.0](http://java.sun.com/javaee/reference/) programming model is the introduction of metadata annotations. Annotations simplify the application development process by allowing a developer to specify within the Java class itself how the application component behaves in the container, requests for dependency injection, and so on. Annotations are an alternative to deployment descriptors that were required by older versions of Enterprise applications (J2EE 1.4 and earlier).

With Java EE annotations, the standard application.xml and web.xml deployment descriptors are optional. The Java EE programming model uses the [JDK 5.0 annotations](http://java.sun.com/javaee/5/docs/api/) feature for Web containers, such as EJBs, servlets, Web applications, and JSPs. See [WebLogic Annotation for Web Components.](https://docs.oracle.com/cd/E13222_01/wls/docs103/webapp/annotateservlet.html)

However, Web applications deployed on WebLogic Server can still use a standard Java EE deployment descriptor file and a WebLogic-specific deployment descriptor file to define their resources and operating attributes.

Web Application Development Key Points

JSPs and HTTP servlets can access all services and APIs available in WebLogic Server. These services include EJBs, database connections by way of Java Database Connectivity (JDBC), Java Messaging Service (JMS), XML, and more.

A Web archive (WAR file) contains the files that make up a Web application. A WAR file is deployed as a unit on one or more WebLogic Server instances. A WAR file deployed to WebLogic Server always includes the following files:

* + One servlet or Java Server Page (JSP), along with any helper classes.
  + An optional web.xml deployment descriptor, which is a Java EE standard XML document that describes the contents of a WAR file.
  + A weblogic.xml deployment descriptor, which is an XML document containing WebLogic Server-specific elements for Web applications.
  + A WAR file can also include HTML or XML pages and supporting files such as image and multimedia files.

The WAR file can be deployed alone or packaged in an Enterprise application archive (EAR file) with other application components. If deployed alone, the archive must end with a .war extension. If deployed in an EAR file, the archive must end with an .ear extension.

Oracle recommends that you package and deploy your stand-alone Web applications as part of an Enterprise application. This is an Oracle best practice, which allows for easier application migration, additions, and changes. Also, packaging your applications as part of an Enterprise application allows you to take advantage of the split development directory structure, which provides a number of benefits over the traditional single directory structure.

|  |  |
| --- | --- |
| **Note:** | If you are deploying a directory in exploded format (not archived), do not name the directory .ear, .jar, and so on. For more information on archived format, see [Web Application Developer Tools](https://docs.oracle.com/cd/E13222_01/wls/docs103/webapp/basics.html#wp136762). |

Servlets

A servlet is a Java class that runs in a Java-enabled server. An HTTP servlet is a special type of servlet that handles an HTTP request and provides an HTTP response, usually in the form of an HTML page. The most common use of WebLogic HTTP Servlets is to create interactive applications using standard Web browsers for the client-side presentation while WebLogic Server handles the business logic as a server-side process. WebLogic HTTP servlets can access databases, Enterprise JavaBeans, messaging APIs, HTTP sessions, and other facilities of WebLogic Server.

Servlets and Java EE

WebLogic Server fully supports HTTP servlets as defined in the [Servlet 2.5 specification](http://java.sun.com/products/servlet/index.jsp) from Sun Microsystems. HTTP servlets form an integral part of the Java EE standard.

With Java EE metadata annotations, the standard web.xml deployment descriptor is now optional. The [Servlet 2.5 specification](http://java.sun.com/products/servlet/index.jsp), states annotations can be defined on certain web components, such as servlets, filters, listeners, and tag handlers. The annotations are used to declare dependencies on external resources. The container will detect annotations on such components and inject necessary dependencies before the component’s life-cycle methods are invoked. See [WebLogic Annotation for Web Components](https://docs.oracle.com/cd/E13222_01/wls/docs103/webapp/annotateservlet.html).

The [Servlet 2.5 specification](http://java.sun.com/products/servlet/index.jsp) defines the implementation of the servlet API and the method by which servlets are deployed in enterprise applications. Deploying servlets on a Java EE-compliant server, such as WebLogic Server, is accomplished by packaging the servlets and other resources that make up an enterprise application into a single unit, the Web application. A Web application utilizes a specific directory structure to contain its resources and a deployment descriptor that defines how these resources interact and how the application is accessed by a client. See [The Web Applications Container](https://docs.oracle.com/cd/E13222_01/wls/docs103/webapp/basics.html#wp108532).

What You Can Do with Servlets

* + Create dynamic Web pages that use HTML forms to get end-user input and provide HTML pages that respond to that input. Examples of this utilization include online shopping carts, financial services, and personalized content.
  + Create collaborative systems such as online conferencing.
  + Have access to a variety of APIs and features by using servlets running in WebLogic Server. For example:
    - Session tracking—Allows a Web site to track a user’s progress across multiple Web pages. This functionality supports Web sites such as e-commerce sites that use shopping carts. WebLogic Server supports session persistence to a database, providing fail-over between server down time and session sharing between clustered servers. For more information see [Session Tracking from a Servlet](https://docs.oracle.com/cd/E13222_01/wls/docs103/webapp/sessions.html#wp150259).
    - JDBC drivers (including Oracle)—JDBC drivers provide basic database access. With WebLogic Server’s multi-tier JDBC implementations, you can take advantage of connection pools, server-side data caching, and transactions. For more information see [Accessing Databases](https://docs.oracle.com/cd/E13222_01/wls/docs103/webapp/progservlet.html#wp159914).
    - Enterprise JavaBeans—Servlets can use Enterprise JavaBeans (EJB) to encapsulate sessions, data from databases, and other functionality. See [Referencing External EJBs](https://docs.oracle.com/cd/E13222_01/wls/docs103/webapp/configureresources.html#wp158714), [More about the ejb-ref\* Elements](https://docs.oracle.com/cd/E13222_01/wls/docs103/webapp/configureresources.html#wp158728), and [Referencing Application-Scoped EJBs](https://docs.oracle.com/cd/E13222_01/wls/docs103/webapp/configureresources.html#wp158740).
    - Java Messaging Service (JMS)—JMS allows your servlets to exchange messages with other servlets and Java programs. See [*Programming WebLogic JMS*](https://docs.oracle.com/cd/E13222_01/wls/docs103/jms/index.html).
    - Java JDK APIs—Servlets can use the standard Java JDK APIs.
    - Forwarding requests—Servlets can forward a request to another servlet or other resource. [Forwarding a Request](https://docs.oracle.com/cd/E13222_01/wls/docs103/webapp/progservlet.html#wp160043).
  + Easily deploy servlets written for any Java EE-compliant servlet engine to WebLogic Server.

Servlet Development Key Points

The following are a few key points relating to servlet development:

* + Programmers of HTTP servlets utilize a standard Java API, javax.servlet.http, to create interactive applications.
  + HTTP servlets can read HTTP headers and write HTML coding to deliver a response to a browser client.
  + Servlets are deployed to WebLogic Server as part of a Web application. A Web application is a grouping of application components such as servlet classes, JavaServer Pages (JSPs), static HTML pages, images, and security.

JavaServer Pages

JavaServer Pages (JSPs) are a Sun Microsystems specification for combining Java with HTML to provide dynamic content for Web pages. When you create dynamic content, JSPs are more convenient to write than HTTP servlets because they allow you to embed Java code directly into your HTML pages, in contrast with HTTP servlets, in which you embed HTML inside Java code. The [JSP 2.1 specification](http://java.sun.com/products/jsp/) is part of the Java Enterprise Edition (Java EE).

JSPs are Web pages coded with an extended HTML that makes it possible to embed Java code in a Web page. JSPs can call custom Java classes, called *taglibs*, using HTML-like tags. The WebLogic appc compiler weblogic.appc generates JSPs and validates descriptors. You can also precompile JSPs into the WEB-INF/classes/ directory or as a JAR file under WEB-INF/lib/ and package the servlet class in the Web archive to avoid compiling in the server. Servlets and JSPs may require additional helper classes to be deployed with the Web application.

JSPs enable you to separate the dynamic content of a Web page from its presentation. It caters to two different types of developers: HTML developers, who are responsible for the graphical design of the page, and Java developers, who handle the development of software to create the dynamic content.

JSPs and Java EE

WebLogic JSP supports the [JSP 2.1 specification](http://java.sun.com/products/jsp/) from Sun Microsystems. The main theme for Java EE is ease of development. The platform’s web tier contributes significantly to ease of development in two ways. First, the platform now includes the JavaServer Pages Standard Tag Library (JSTL) and JavaServer Faces technology. Second, all the web-tier technologies offer a set of features that make development of web applications on Java EE much easier, such as:

* + A new expression language (EL) syntax that allows deferred evaluation of expressions, enables using expressions to both get and set data and to invoke methods, and facilitates customizing the resolution of a variable or property referenced by an expression.
  + Support for resource injection through annotations to simplify configuring access to resources and environment data.
  + Complete alignment of JavaServer Faces technology tags and JavaServer Pages (JSP) software code.

Because JSPs are part of the Java EE standard, you can deploy JSPs on a variety of platforms, including WebLogic Server. In addition, third-party vendors and application developers can provide JavaBean components and define custom JSP tags that can be referenced from a JSP page to provide dynamic content.

What You Can Do with JSPs

* + Combine Java with HTML to provide dynamic content for Web pages.
  + Call custom Java classes, called *taglibs*, using HTML-like tags.
  + Embed Java code directly into your HTML pages, in contrast with HTTP servlets, in which you embed HTML inside Java code.
  + Separate the dynamic content of a Web page from its presentation.

Overview of How JSP Requests Are Handled

WebLogic Server handles JSP requests in the following sequence:

* 1. A browser requests a page with a .jsp file extension from WebLogic Server.
  2. WebLogic Server reads the request.
  3. Using the JSP compiler, WebLogic Server converts the JSP into a servlet class that implements the javax.servlet.jsp.JspPage interface. The JSP file is compiled only when the page is first requested, or when the JSP file has been changed. Otherwise, the previously compiled JSP servlet class is re-used, making subsequent responses much quicker.
  4. The generated JspPage servlet class is invoked to handle the browser request.

It is also possible to invoke the JSP compiler directly without making a request from a browser. For details, see [Using the WebLogic JSP Compiler](https://docs.oracle.com/cd/E13222_01/wls/docs103/webapp/reference.html#wp57794).

Because the JSP compiler creates a Java servlet as its first step, you can look at the Java files it produces, or even register the generated JspPage servlet class as an HTTP servlet. See [Servlets](https://docs.oracle.com/cd/E13222_01/wls/docs103/webapp/basics.html#wp152236).

Web Application Developer Tools

Oracle provides several tools to help you create and configure Web applications. These are discussed in the following sections.

Ant Tasks to Create Skeleton Deployment Descriptors

You can use the WebLogic Ant utilities to create skeleton deployment descriptors. These utilities are Java classes shipped with your WebLogic Server distribution. The Ant task looks at a directory containing a Web application and creates deployment descriptors based on the files it finds in the Web application. Because the Ant utility does not have information about all desired configurations and mappings for your Web application, the skeleton deployment descriptors the utility creates are incomplete. After the utility creates the skeleton deployment descriptors, you can use a text editor, an XML editor, or the Administration Console to edit the deployment descriptors and complete the configuration of your Web application.

XML Editors

You can use an enterprise-level IDE with DTD validation or another development tool that supports editing of XML files.

Web Application Security

You can secure a Web application by restricting access to certain URL patterns in the Web application or programmatically using security calls in your servlet code.

At runtime, your username and password are authenticated using the applicable  
security realm for the Web application. Authorization is verified according to the security constraints configured in web.xml or the external policies that might have been created using Administration Console for the Web application.

At runtime, the WebLogic Server active security realm applies the Web application security constraints to the specified Web application resources. Note that a security realm is shared across multiple virtual hosts.

For detailed instructions and an example on configuring security in Web applications, see [Securing WebLogic Resources](https://docs.oracle.com/cd/E13222_01/wls/docs103/secwlres/index.html). For more information on WebLogic security, refer to [*Programming WebLogic Security*](https://docs.oracle.com/cd/E13222_01/wls/docs103/security/index.html).

P3P Privacy Protocol

The Platform for Privacy Preferences (P3P) provides a way for Web sites to publish their privacy policies in a machine-readable syntax. The WebLogic Server Web application container can support P3P.

There are three ways to tell the browser about the location of the p3p.xml file:

* + Place the a policy reference file in the “well-known location” (at the location /w3c/p3p.xml on the site).
  + Add an extra HTTP header to each response from the Web site giving the location of the policy reference file.
  + Place a link to the policy reference file in each HTML page on the site.

For more detailed information, see [http://www.w3.org/TR/p3pdeployment#Locating\_PRF.](http://www.w3.org/TR/p3pdeployment#Locating_PRF)

Displaying Special Characters on Linux Browsers

To display special characters on Linux browsers, set the JVM’s file.encoding system property to ISO8859\_1. For example, java -Dfile.encoding=ISO8859\_1 weblogic.Server. For a complete listing, see Sun’s [Supported Encodings](http://java.sun.com/j2se/1.5.0/docs/guide/intl/encoding.doc.html) page for Java EE 5.0.