web.xml Deployment Descriptor Elements

The following sections describe the deployment descriptor elements defined in the `web.xml` schema under the root element `<web-app>`.

With Java EE annotations, the standard `web.xml` deployment descriptor is optional. According to the Servlet 2.5 specification, 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. See Chapter 8, "WebLogic Annotation for Web Components."

- "jsp-config" on page A-14
- "listener" on page A-8
- "login-config" on page A-22
- "mime-mapping" on page A-13
- "resource-env-ref" on page A-17
- "resource-ref" on page A-18
- "security-constraint" on page A-19
- "security-role" on page A-23
- "servlet" on page A-9
- "servlet-mapping" on page A-11
- "session-config" on page A-12
- "web-app" on page A-26
- "welcome-file-list" on page A-13

web.xml Namespace Declaration and Schema Location

The correct text for the namespace declaration and schema location for the web.xml file is as follows.

```
<web-app xmlns="http://java.sun.com/xml/ns/j2ee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" version="2.5">
```

To view the schema for `web.xml`, go to http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd.

icon

The `icon` element specifies the location within the Web application for a small and large image used to represent the Web application in a GUI tool. (The `servlet` element also has an element called the `icon` element, used to supply an icon to represent a servlet in a GUI tool.)

The following table describes the elements you can define within an `icon` element.

Element	Required/ Optional	Description
`<small-icon>`	Optional	Location for a small (16x16 pixel) `.gif` or `.jpg` image used to represent the Web application in a GUI tool. Currently, this is not used by WebLogic Server.
`<large-icon>`	Optional	Location for a large (32x32 pixel) `.gif` or `.jpg` image used to represent the Web application in a GUI tool. Currently, this element is not used by WebLogic Server.

display-name

The optional `display-name` element specifies the Web application display name, a short name that can be displayed by GUI tools.

Element	Required/ Optional	Description
`<display-name>`	Optional	Currently, this element is not used by WebLogic Server.

description

The optional description element provides descriptive text about the Web application.

Element	Required/ Optional	Description
`<description>`	Optional	Currently, this element is not used by WebLogic Server.

distributable

The distributable element is not used by WebLogic Server.

Element	Required/ Optional	Description
`<distributable>`	Optional	Currently, this element is not used by WebLogic Server.

context-param

The optional `context-param` element contains the declaration of a Web application's servlet context initialization parameters. The following table describes the reserved context parameters used by the Web application container, which have been deprecated and have replacements in weblogic.xml.

Deprecated Parameter	Description	Replacement Element in weblogic.xml
`weblogic.httpd.inputCharset`	Defines code set behavior for non-unicode operations.	`input-charset` (defined within charset-param) in weblogic.xml. See "input-charset" on page B-21.
`weblogic.httpd.servlet.reloadCheckSecs`	Define how often WebLogic Server checks whether a servlet has been modified, and if so, reloads it. A value of -1 is never reload, 0 is always reload. The default is set to 1 second.	`servlet-reload-check-secs` (defined within `container-descriptor`) in `weblogic.xml`.See "container-descriptor" on page B-16.

Deprecated Parameter	Description	Replacement Element in weblogic.xml
`weblogic.httpd.servlet.classpath`	When this values has been set, the container appends this path to the Web application classpath. This is not a recommended method and is supported only for backward compatibility.	No replacement. Use other means such as manifest classpath or `WEB-INF/lib` or `WEB-INF/classes` or virtual directories.
`weblogic.httpd.defaultServlet`	Sets the default servlet for the Web application. This is not a recommended method and is supported only for backward compatibility.	No replacement. Instead use the `servlet` and `servlet-mapping` elements in `web.xml` to define a default servlet. The URL pattern for `default-servlet` should be "/". See "servlet-mapping" on page A-11. For additional examples of servlet mapping, see "Servlet Mapping" on page 4-2.

The following `context-param` parameter is still valid.

Element	Required/ Optional	Description
`weblogic.httpd.clientCertProxy`	optional	This attribute specifies that certifications from clients of the Web application are provided in the special `WL-Proxy-Client-Cert` header sent by a proxy plug-in or `HttpClusterServlet`. This setting is useful if user authentication is performed on a proxy server—setting `clientCertProxy` causes the proxy server to pass on the certs to the cluster in a special header, `WL-Proxy-Client-Cert`. A `WL-Proxy-Client-Cert` header could be provided by any client with access to WebLogic Server. WebLogic Server takes the certificate information from that header, trusting that is came from a secure source (the plug-in) and uses that information to authenticate the user. For this reason, if you set `clientCertProxy`, use a connection filter to ensure that WebLogic Server accepts connections only from the machine on which the plug-in is running. In addition to setting this attribute for an individual Web application, you can define this attribute: • For all web applications hosted by a server instance, on the Server-->Configuration-->General page in the Administration Console • For all web applications hosted by server instances in a cluster, on the Cluster-->Configuration-->General page.

filter

The `filter` element defines a filter class and its initialization attributes. For more information on filters, see “Configuring Filters” on page 14-3.

The following table describes the elements you can define within a `filter` element.

Element	Required/ Optional	Description
`<icon>`	Optional	Specifies the location within the Web application for a small and large image used to represent the filter in a GUI tool. Contains a small-icon and large-icon element. Currently, this element is not used by WebLogic Server.
`<filter-name>`	Required	Defines the name of the filter, used to reference the filter definition elsewhere in the deployment descriptor.
`<display-name>`	Optional	A short name intended to be displayed by GUI tools.
`<description>`	Optional	A text description of the filter.
`<filter-class>`	Required	The fully-qualified class name of the filter.
`<init-param>`	Optional	Contains a name/value pair as an initialization attribute of the filter. Use a separate set of `<init-param>` tags for each attribute.

filter-mapping

The following table describes the elements you can define within a `filter-mapping` element.

Element	Required/ Optional	Description
`<filter-name>`	Required	The name of the filter to which you are mapping a URL pattern or servlet. This name corresponds to the name assigned in the `<filter>` element with the `<filter-name>` element.

`<url-pattern>`	Required - or map by `<servlet>`	Describes a pattern used to resolve URLs. The portion of the URL after the `http://host:port` + `ContextPath` is compared to the `<url-pattern>` by WebLogic Server. If the patterns match, the filter mapped in this element is called.
		Example patterns:
		`/soda/grape/*`
		`/foo/*`
		`/contents`
		`*.foo`
		The URL must follow the rules specified in the Servlet 2.3 Specification.

<servlet>	Required - or map by `<url-pattern>`	The name of a servlet which, if called, causes this filter to execute.

listener

Define an application listener using the `listener` element.

Element	Required/ Optional	Description
`<listener-class>`	Optional	Name of the class that responds to a Web application event.

For more information, see “Configuring an Event Listener Class” on page 11-4.

servlet

The `servlet` element contains the declarative data of a servlet.

If a `jsp-file` is specified and the `<load-on-startup>` element is present, then the JSP is precompiled and loaded when WebLogic Server starts.

The following table describes the elements you can define within a `servlet` element.

Element	Required/ Optional	Description
`<icon>`	Optional	Location within the Web application for a small and large image used to represent the servlet in a GUI tool. Contains a small-icon and large-icon element. Currently, this element is not used by WebLogic Server.
`<servlet-name>`	Required	Defines the canonical name of the servlet, used to reference the servlet definition elsewhere in the deployment descriptor.
`<display-name>`	Optional	A short name intended to be displayed by GUI tools.
`<description>`	Optional	A text description of the servlet.
`<servlet-class>`	Required (or use `<jsp-file>`)	The fully-qualified class name of the servlet. Use only one of either the `<servlet-class>` tags or `<jsp-file>` tags in your servlet body.
`<jsp-file>`	Required (or use `<servlet-class>`)	The full path to a JSP file within the Web application, relative to the Web application root directory. Use only one of either the `<servlet-class>` tags or `<jsp-file>` tags in your servlet body.
`<init-param>`	Optional	Contains a name/value pair as an initialization attribute of the servlet. Use a separate set of `<init-param>` tags for each attribute.
`<load-on-startup>`	Optional	WebLogic Server initializes this servlet when WebLogic Server starts up. The optional content of this element must be a positive integer indicating the order in which the servlet should be loaded. Lower integers are loaded before higher integers. If no value is specified, or if the value specified is not a positive integer, WebLogic Server can load the servlet in any order during application startup.

Element	Required/ Optional	Description
`<run-as>`	Optional	Specifies the run-as identity to be used for the execution of the Web application. It contains an optional description and the name of a security role.
`<security-role-ref>`	Optional	Used to link a security role name defined by `<security-role>` to an alternative role name that is hard coded in the servlet logic. This extra layer of abstraction allows the servlet to be configured at deployment without changing servlet code.

icon

This is an element within the "servlet" on page A-9.

The `icon` element specifies the location within the Web application for small and large images used to represent the servlet in a GUI tool.

The following table describes the elements you can define within an `icon` element.

Element	Required/ Optional	Description
`<small-icon>`	Optional	Specifies the location within the Web application for a small (16x16 pixel) `.gif` or `.jpg` image used to represent the servlet in a GUI tool. Currently, this element is not used by WebLogic Server.
`<large-icon>`	Optional	Specifies the location within the Web application for a small (32x32 pixel) `.gif` or `.jpg` image used to represent the servlet in a GUI tool. Currently, this element is not used by WebLogic Server.

init-param

This is an element within the "servlet" on page A-9.

The optional `init-param` element contains a name/value pair as an initialization attribute of the servlet. Use a separate set of `init-param` tags for each attribute.

You can access these attributes with the `javax.servlet.ServletConfig.getInitParameter()` method.

The following table describes the elements you can define within a `init-param` element.

Element	Required/ Optional	Description
`<param-name>`	Required	Defines the name of this attribute.
`<param-value>`	Required	Defines a `String` value for this attribute.
`<description>`	Optional	Text description of the initialization attribute.

security-role-ref

This is an element within the "servlet" on page A-9.

The `security-role-ref` element links a security role name defined by `<security-role>` to an alternative role name that is hard-coded in the servlet logic. This extra layer of abstraction allows the servlet to be configured at deployment without changing servlet code.

The following table describes the elements you can define within a `security-role-ref` element.

Element	Required/ Optional	Description
`<description>`	Optional	Text description of the role.
`<role-name>`	Required	Defines the name of the security role or principal that is used in the servlet code.
`<role-link>`	Required	Defines the name of the security role that is defined in a `<security-role>` element later in the deployment descriptor.

servlet-mapping

The `servlet-mapping` element defines a mapping between a servlet and a URL pattern.

The following table describes the elements you can define within a `servlet-mapping` element.

Element	Required/ Optional	Description
`<servlet-name>`	Required	The name of the servlet to which you are mapping a URL pattern. This name corresponds to the name you assigned a servlet in a `<servlet>` declaration tag.

`<url-pattern>`	Required	Describes a pattern used to resolve URLs. The portion of the URL after the `http://host:port` + `WebAppName` is compared to the `<url-pattern>` by WebLogic Server. If the patterns match, the servlet mapped in this element will be called.
		
		Example patterns:
		
		`/soda/grape/*`
		`/foo/*`
		`/contents`
		`*.foo`
		
		The URL must follow the rules specified in the Servlet 2.3 Specification.
		
		For additional examples of servlet mapping, see "Servlet Mapping" on page 4-2.

session-config

The `session-config` element defines the session attributes for this Web application.

The following table describes the element you can define within a `session-config` element.

Element	Required/ Optional	Description
`<session-timeout>`	Optional	The number of minutes after which sessions in this Web application expire. The value set in this element overrides the value set in the `TimeoutSecs` attribute of the `<session-descriptor>` element in the WebLogic-specific deployment descriptor `weblogic.xml`, unless one of the special values listed here is entered. Default value: 60 Maximum value: Integer.MAX_VALUE ÷ 60 Special values: • -1 = Sessions do not timeout. The value set in `<session-descriptor>` element of `weblogic.xml` is ignored. For more information, see "session-descriptor" on page B-7.

mime-mapping

The `mime-mapping` element defines a mapping between an extension and a mime type.

The following table describes the elements you can define within a `mime-mapping` element.

Element	Required/ Optional	Description
`<extension>`	Required	A string describing an extension, for example: `txt.`
`<mime-type>`	Required	A string describing the defined mime type, for example: `text/plain.`

welcome-file-list

The optional `welcome-file-list` element contains an ordered list of `welcome-file` elements.

When the URL request is a directory name, WebLogic Server serves the first file specified in this element. If that file is not found, the server then tries the next file in the list.

For more information, see "Configuring Welcome Files" on page 5-4.

The following table describes the element you can define within a `welcome-file-list` element.

Element	Required/ Optional	Description
`<welcome-file>`	Optional	File name to use as a default welcome file, such as index.html

error-page

The optional `error-page` element specifies a mapping between an error code or exception type to the path of a resource in the Web application.

When an error occurs—while WebLogic Server is responding to an HTTP request, or as a result of a Java exception—WebLogic Server returns an HTML page that displays either the HTTP error code or a page containing the Java error message. You can define your own HTML page to be displayed in place of these default error pages or in response to a Java exception.

For more information, see "Customizing HTTP Error Responses" on page 5-5.

The following table describes the elements you can define within an `error-page` element.

Note: Define either an `<error-code>` or an `<exception-type>` but not both.

Element	Required/ Optional	Description
`<error-code>`	Optional	A valid HTTP error code, for example, `404.`
`<exception-type>`	Optional	A fully-qualified class name of a Java exception type, for example, `java.lang.string`
`<location>`	Required	The location of the resource to display in response to the error. For example, `/myErrorPg.html.`

jsp-config

The `jsp-config` element is used to provide global configuration information for the JSP files in a Web application. It has two sub-elements, `taglib` and `jsp-property-group`.

The following table describes the elements you can define within a `jsp-config` element.

Element	Required/ Optional	Description
`<taglib>`	Optional	Provides information on a tag library that is used by a JSP page within the Web application.
`<jsp-property-group>`	Optional	Used to group a number of files so they can be given global property information. All files so described are deemed to be JSP files.

taglib

This is an element within the "jsp-config" on page A-14.

The required `taglib` element provides information on a tag library that is used by a JSP page within the Web application.

This element associates the location of a JSP Tag Library Descriptor (TLD) with a URI pattern. Although you can specify a TLD in your JSP that is relative to the `WEB-INF` directory, you can also use the `<taglib>` tag to configure the TLD when deploying your Web application. Use a separate element for each TLD.

The following table describes the elements you can define within a `taglib` element.

Element	Required/ Optional	Description
`<taglib-location>`	Optional	Gives the file name of the tag library descriptor relative to the root of the Web application. It is a good idea to store the tag library descriptor file under the `WEB-INF` directory so it is not publicly available over an HTTP request.
`<taglib-uri>`	Optional	Describes a URI, relative to the location of the web.xml document, identifying a Tag Library used in the Web application. If the URI matches the URI string used in the taglib directive on the JSP page, this taglib is used.

jsp-property-group

This is an element within the "jsp-config" on page A-14.

The required `jsp-property-group` element is used to group a number of files so they can be given global property information. All files so described are deemed to be JSP files.

The following table describes the elements you can define within a `jsp-property-group` element.

Element	Required/ Optional	Description
`<el-ignored>`	Optional	Controls whether EL is ignored. By default, the EL evaluation is enabled for Web Applications using a Servlet 2.4 or greater `web.xml`, and disabled otherwise.
`<scripting-invalid>`	Optional	Controls whether scripting elements are invalid in a group of JSP pages. By default, scripting is enabled.
`<page-encoding>`	Optional	Indicates pageEncoding information. It is a translation-time error to name different encodings in the pageEncoding attribute of the page directive of a JSP page and in a JSP configuration element matching the page. It is also a translation-time error to name different encodings in the prolog or text declaration of a document in XML syntax and in a JSP configuration element matching the document. It is legal to name the same encoding through multiple mechanisms.
`<is-xml>`	Optional	Indicates that a resource is a JSP document (XML). If true, denotes that the group of resources that match the URL pattern are JSP documents, and thus must be interpreted as XML documents. If false, the resources are assumed to not be JSP documents, unless there is another property group that indicates otherwise.
`<include-prelude>`	Optional	A context-relative path that must correspond to an element in the Web Application. When the element is present, the given path will be automatically included (as in an include directive) at the beginning of each JSP page in this jsp-property-group.
`<include-coda>`	Optional	A context-relative path that must correspond to an element in the Web application. When the element is present, the given path will be automatically included (as in an include directive) at the end of each JSP page in this `jsp-property-group`.
`<deferred-syntax-allowed-as-literal>`	Optional	Controls whether the character sequence `#{` is allowed when used as a String literal.

Element	Required/ Optional	Description
`<trim-directive-whitespaces>`	Optional	Controls whether template text containing only white spaces must be removed from the response output.

`<url-pattern>`	Required	Describes a pattern used to resolve URLs. The portion of the URL after the `http://host:port` + `ContextPath` is compared to the `<url-pattern>` by WebLogic Server.
		Example patterns:
		`/soda/grape/*`
		`/foo/*`
		`/contents`
		`*.foo`
		The URL must follow the rules specified in the Servlet 2.4 Specification.

resource-env-ref

The `resource-env-ref` element contains a declaration of a Web application's reference to an administered object associated with a resource in the Web application's environment. It consists of an optional description, the resource environment reference name, and an indication of the resource environment reference type expected by the Web application code.

For example:

```
<resource-env-ref>

    <resource-env-ref-name>jms/StockQueue</resource-env-ref-name>

    <resource-env-ref-type>javax.jms.Queue</resource-env-ref-type>

</resource-env-ref>
```

The following table describes the elements you can define within a `resource-env-ref` element.

Element	Required/ Optional	Description
`<description>`	Optional	Provides a description of the resource environment reference.
`<resource-env-ref -name>`	Required	Specifies the name of a resource environment reference; its value is the environment entry name used in the Web application code. The name is a JNDI name relative to the `java:comp/env` context and must be unique within a Web application.
<resource-env-ref-type>	Required	Specifies the type of a resource environment reference. It is the fully qualified name of a Java language class or interface.

resource-ref

The optional `resource-ref` element defines a reference lookup name to an external resource. This allows the servlet code to look up a resource by a “virtual” name that is mapped to the actual location at deployment time.

Use a separate `<resource-ref>` element to define each external resource name. The external resource name is mapped to the actual location name of the resource at deployment time in the WebLogic-specific deployment descriptor `weblogic.xml`.

The following table describes the elements you can define within a `resource-ref` element.

Element	Required/ Optional	Description
`<description>`	Optional	A text description.
`<res-ref-name>`	Required	The name of the resource used in the JNDI tree. Servlets in the Web application use this name to look up a reference to the resource.
`<res-type>`	Required	The Java type of the resource that corresponds to the reference name. Use the full package name of the Java type.

Element	Required/ Optional	Description
`<res-auth>`	Required	Used to control the resource sign on for security. If set to `APPLICATION,` indicates that the application component code performs resource sign on programmatically. If set to `CONTAINER`, WebLogic Server uses the security context established with the `login-config` element. See "login-config" on page A-22.
`<res-sharing-scope>`	Optional	Specifies whether connections obtained through the given resource manager connection factory reference can be shared. Valid values: • Shareable • Unshareable

security-constraint

The `security-constraint` element defines the access privileges to a collection of resources defined by the `<web-resource-collection>` element.

For detailed instructions and an example on configuring security in Web applications, see *Securing WebLogic Resources*. Also, for more information on WebLogic Security, refer to *Programming WebLogic Security*.

The following table describes the elements you can define within a `security-constraint` element.

Element	Required/ Optional	Description
`<web-resource-collection>`	Required	Defines the components of the Web application to which this security constraint is applied.
`<auth-constraint>`	Optional	Defines which groups or principals have access to the collection of web resources defined in this security constraint. See also "auth-constraint" on page A-20.
`<user-data-constraint>`	Optional	Defines how the client should communicate with the server. See also "user-data-constraint" on page A-21.

web-resource-collection

Each `<security-constraint>` element must have one or more `<web-resource-collection>` elements. These define the area of the Web application to which this security constraint is applied.

This is an element within the "security-constraint" on page A-19.

The following table describes the elements you can define within a `web-resource-collection` element.

Element	Required/ Optional	Description
`<web-resource-name>`	Required	The name of this Web resource collection.
`<description>`	Optional	A text description of this security constraint.
`<url-pattern>`	Optional	Use one or more of the `<url-pattern>` elements to declare to which URL patterns this security constraint applies. If you do not use at least one of these elements, this `<web-resource-collection>` is ignored by WebLogic Server.
`<http-method>`	Optional	Use one or more of the `<http-method>` elements to declare which HTTP methods (usually, `GET` or `POST`) are subject to the authorization constraint. If you omit the `<http-method>` element, the default behavior is to apply the security constraint to all HTTP methods.

auth-constraint

This is an element within the "security-constraint" on page A-19.

The optional `auth-constraint` element defines which groups or principals have access to the collection of Web resources defined in this security constraint.

The following table describes the elements you can define within an `auth-constraint` element.

Element	Required/ Optional	Description
`<description>`	Optional	A text description of this security constraint.
`<role-name>`	Optional	Defines which security roles can access resources defined in this security-constraint. Security role names are mapped to principals using the `security-role-ref`. See "security-role-ref" on page A-11.

user-data-constraint

This is an element within the "security-constraint" on page A-19.

The `user-data-constraint` element defines how the client should communicate with the server.

The following table describes the elements you may define within a `user-data-constraint` element.

Element	Required/ Optional	Description
`<description>`	Optional	A text description.
`<transport-guarantee>`	Required	Specifies that the communication between client and server. WebLogic Server establishes a Secure Sockets Layer (SSL) connection when the user is authenticated using the `INTEGRAL` or `CONFIDENTIAL` transport guarantee. Range of values: • `NONE`–The application does not require any transport guarantees. • `INTEGRAL`–The application requires that the data be sent between the client and server in such a way that it cannot be changed in transit. • `CONFIDENTIAL`–The application requires that data be transmitted so as to prevent other entities from observing the contents of the transmission.

login-config

Use the optional `login-config` element to configure how the user is authenticated; the realm name that should be used for this application; and the attributes that are needed by the form login mechanism.

If this element is present, the user must be authenticated in order to access any resource that is constrained by a `<security-constraint>` defined in the Web application. Once authenticated, the user can be authorized to access other resources with access privileges.

The following table describes the elements you can define within a `login-config` element.

Element	Required/ Optional	Description
`<auth-method>`	Optional	Specifies the method used to authenticate the user. Possible values: `BASIC`—uses browser authentication. (This is the default value.) `FORM`—uses a user-written HTML form. `CLIENT-CERT` **Note:** You can define multiple authentication methods as a comma separated list to provide a fall-back mechanism. Authentication will be attempted in the order the values are defined in the `auth-method` list. See Providing a Fallback Mechanism for Authentication Methods in *Programming WebLogic Security*.
`<realm-name>`	Optional	The name of the realm that is referenced to authenticate the user credentials. If omitted, the realm defined with the Auth Realm Name field on the Web application→ Configuration→Other tab of the Administration Console is used by default. **Note:** The `<realm-name>` element does not refer to system security realms within WebLogic Server. This element defines the realm name to use in HTTP Basic authorization. The system security realm is a collection of security information that is checked when certain operations are performed in the server. The servlet security realm is a different collection of security information that is checked when a page is accessed and basic authentication is used.
`<form-login-config>`	Optional	Use this element if you configure the `<auth-method>` to `FORM`. See "form-login-config" on page A-23.

form-login-config

This is an element within the "login-config" on page A-22.

Use the `<form-login-config>` element if you configure the `<auth-method>` to `FORM`.

.

Element	Required/ Optional	Description
`<form-login-page>`	Required	The URI of a Web resource relative to the document root, used to authenticate the user. This can be an HTML page, JSP, or HTTP servlet, and must return an HTML page containing a `FORM`-based authentication that conforms to a specific naming convention.
`<form-error-page>`	Required	The URI of a Web resource relative to the document root, sent to the user in response to a failed authentication login.

security-role

The following table describes the elements you can define within a `security-role` element.

Element	Required/ Optional	Description
`<description>`	Optional	A text description of this security role.
`<role-name>`	Required	The role name. The name you use here must have a corresponding entry in the WebLogic-specific deployment descriptor, `weblogic.xml,` which maps roles to principals in the security realm. For more information, see "security-role-assignment" on page B-3.

env-entry

The optional `env-entry` element declares an environment entry for an application. Use a separate element for each environment entry.

The following table describes the elements you can define within an `env-entry` element.

Element	Required/ Optional	Description
`<description>`	Optional	A textual description.
`<env-entry-name>`	Required	The name of the environment entry.
`<env-entry-value>`	Required	The value of the environment entry.

`<env-entry-type>`	Required	The type of the environment entry.
		Can be set to one of the following Java types:
		`java.lang.Boolean`
		`java.lang.String`
		`java.lang.Integer`
		`java.lang.Double`
		`java.lang.Float`

ejb-ref

The optional `ejb-ref` element defines a reference to an EJB resource. This reference is mapped to the actual location of the EJB at deployment time by defining the mapping in the WebLogic-specific deployment descriptor file, `weblogic.xml`. Use a separate `<ejb-ref>` element to define each reference EJB name.

The following table describes the elements you can define within an `ejb-ref` element.

Element	Required/ Optional	Description
`<description>`	Optional	A text description of the reference.
`<ejb-ref-name>`	Required	The name of the EJB used in the Web application. This name is mapped to the JNDI tree in the WebLogic-specific deployment descriptor `weblogic.xml`. For more information, see "ejb-reference-description" on page B-6.
`<ejb-ref-type>`	Required	The expected Java class type of the referenced EJB.
`<home>`	Required	The fully qualified class name of the EJB home interface.

Element	Required/ Optional	Description
`<remote>`	Required	The fully qualified class name of the EJB remote interface.
`<ejb-link>`	Optional	The `<ejb-name>` of an EJB in an encompassing J2EE application package.
`<run-as>`	Optional	A security role whose security context is applied to the referenced EJB. Must be a security role defined with the `<security-role>` element.

ejb-local-ref

The `ejb-local-ref` element is used for the declaration of a reference to an enterprise bean's local home. The declaration consists of:

- An optional description
- The EJB reference name used in the code of the Web application that references the enterprise bean. The expected type of the referenced enterprise bean
- The expected local home and local interfaces of the referenced enterprise bean
- Optional ejb-link information, used to specify the referenced enterprise bean

The following table describes the elements you can define within an `ejb-local-ref` element.

Element	Required/ Optional	Description
`<description>`	Optional	A text description of the reference.
`<ejb-ref-name>`	Required	Contains the name of an EJB reference. The EJB reference is an entry in the Web application's environment and is relative to the `java:comp/env` context. The name must be unique within the Web application. It is recommended that name is prefixed with `ejb/`. For example: `<ejb-ref-name>ejb/Payroll</ejb-ref-name>`

Element	Required/ Optional	Description

`<ejb-ref-type>`	Required	The `ejb-ref-type` element contains the expected type of the referenced enterprise bean. The `ejb-ref-type` element must be one of the following:
		`<ejb-ref-type>Entity</ejb-ref-type>`
		`<ejb-ref-type>Session</ejb-ref-type>`

`<local-home>`	Required	Contains the fully-qualified name of the enterprise bean's local home interface.
`<local>`	Required	Contains the fully-qualified name of the enterprise bean's local interface.

`<ejb-link>`	Optional	The `ejb-link` element is used in the `ejb-ref` or `ejb-local-ref` elements to specify that an EJB reference is linked to an EJB.
		The name in the `ejb-link` element is composed of a path name. This path name specifies the `ejb-jar` containing the referenced EJB with the `ejb-name` of the target bean appended and separated from the path name by #.
		The path name is relative to the WAR file containing the Web application that is referencing the EJB. This allows multiple EJBs with the same `ejb-name` to be uniquely identified.
		Used in: `ejb-local-ref` and `ejb-ref` elements.
		Examples:
		`<ejb-link>EmployeeRecord</ejb-link>`
		`<ejb-link>../products/product.jar#ProductEJB</ejb-link>`

web-app

The XML Schema for the Servlet 2.4 deployment descriptor. WebLogic Server fully supports HTTP servlets as defined in the Servlet 2.4 specification from Sun Microsystems. However, the `version` attributed must be set to `2.4` in order to enforce 2.4 behavior.

The following table describes the elements you can define within an `web-app` element.

Element	Required/ Optional	Description
`<version>`	Required	All Servlet deployment descriptors must indicate the 2.4 version of the schema in order to enforce Servlet 2.4 behavior.