# Glossary

**Absolute Path** A path describing where a file is located, described from the root of the file system. The ServletContext object has a method called `getRealPath()` that returns an absolute path from a partial path pointing to a resource within a web application.

**Absolute URL** A URL that fully describes the path to a resource, including protocol and host information—for example, `http://mywebhost:9080/mywebapp/myresource.jsp`.

**Action** JavaServer Pages contain actions that are elements in the page source expressed in XML syntax. During translation of the JavaServer Page, an action is transformed into Java source code placed in the `_jspService()` method of the generated servlet. An action has an opening tag, closing tag, and body (in some cases, the body is absent). Actions are grouped into standard actions (which must be supported by a J2EE-compliant JSP container) and custom actions (which share the syntax and mechanisms of standard actions, but are optionally present within a JSP container or web application). Custom actions are frequently referred to as custom tags.

***application*** An implicit variable, available within JavaServer Page source, which represents the ServletContext object for a web application.

**Application Scope** The available lifetime for attributes associated with an entire web application—from the point where a web application starts up to the point where the web application is stopped (these points often coincide with application server startup and shutdown).

**Application Server** A body of software that is capable of running J2EE applications. An application server can be expected to have at least the following features:

- The ability to respond to different kinds of requests from different kinds of clients
- The ability to return responses
- A web container for housing web applications
- An EJB container for providing life cycle support to Enterprise JavaBeans
- Other services: naming support (JNDI), database access support (JDBC), security support, and so on

Commercial application servers include IBM WebSphere and BEA WebLogic. Open-source application servers include Tomcat, JBoss, and Apache Geronimo.

**Attribute** A piece of information held as a name and a value. Attributes occur in two main places in web applications:

1. When attaching information to any of the four scopes in a web application: page, request, session, or application. Methods exist to read and write attributes that comprise a name (any String) and a value (any kind of Object).
2. Within XML (including the XML used to describe JSP actions) and HTML, attributes are frequently found in the opening tags of elements. These take the form name="value" (for example, `<c:set var="myVariable">myValue</c:set>`).

**Authentication** The process of validating a "principal" to a computer system. Principal is a general term meaning either a human user or another computer system. The principal supplies credentials to the security component of the target computer system, which accepts or rejects them according to its internal rules. Web applications have four types of authentication available: BASIC, FORM, DIGEST, and CLIENT-CERT.

**Authorization** After authentication has taken place, the process of authorization determines which resources a principal is allowed to use within a computer system. Web applications allow administrators to associate target resources with given roles. If a principal belongs to the given role, access is granted to the protected resource.

**BASIC** A form of authentication. BASIC authentication operates at the point where a principal first makes a request for a protected resource within a web application. The application server returns an HTTP response demanding credentials in the form of a user ID and password (typically, a browser will pop up a dialog box for the user to type these in). The user ID and password are only weakly encoded when transmitted back to the application server. Consequently, BASIC authentication should be used in conjunction with SSL.

**Body** In an HTML element or XML element (including JSP actions), the part that comes between the opening and closing tag. Here is an HTML example: `<p>`**This is the body of the paragraph element in HTML**`</p>`.

**Business Delegate** A design pattern that suggests providing a layer between presentation components and business components. This layer can perform several useful functions such as dealing with complex business component APIs, and their exceptions; simplifying the interface presentation code needs to make use of business services; and implementing retry and recovery code in the event of business service failure.

**CGI** *See* **Common Gateway Interface**.

**Classic Tag Handler** The original JavaServer Page standard for tag handler code. Classic tag handlers have a relatively complex life cycle, with a number of methods called by the JSP container in response to events such as processing the opening tag, the body, and the closing tag.

**CLIENT-CERT** A very secure form of authentication. On requesting a secure resource in a web application, the application server demands a digital certificate from the client. If the client returns a digital certificate that matches those known to the application server, authentication is granted.

**Common Gateway Interface (CGI)** One of the original standards for providing dynamic content within web pages. The standard defines how a World Wide Web server can interact with a program. Typically, the result of running such a program is to return data within web page output. Java web applications represent an improvement on CGI in performance and maintainability.

***config*** An implicit object available within a JavaServer page of ServletConfig type, capable, for example, of returning initialization parameters associated with the JavaServer Page.

**Container** A piece of software that provides life cycle support to J2EE components. An application server typically contains several containers: a web container to run servlets and JavaServer Pages and an EJB container to run Enterprise JavaBeans.

**Context** The part of a URL that uniquely separates one web application from another installed on the same web application server.

**Context Root** The location of the top-level directory that houses one individual web application.

**Cookie** A short file (usually just a string of text) that can accompany an HTTP request or HTTP response. Used to pass information between clients and servers in web applications. Often, this information is for tracking purposes—to attach a session ID to a series of unrelated requests, for example.

**Custom Action** *See* **Action**.

**Custom Tag** *See* **Action**.

**Declaration** In a JavaServer Page, a declaration is a piece of code that contains complete method definitions (and—though not recommended—class or instance data members). The code is included within the servlet source file generated from the JavaServer Page source.

**Declarative Security** Security that is imposed by declaring information within a web application's deployment descriptor, rather than in program code.

**DELETE** This HTTP method deletes a resource at a target URL. Understandably, most web servers disallow execution of this method.

**Deployment Descriptor** A file used within a J2EE module to define its characteristics. For a web application, the deployment descriptor defines resource (servlet) names, resource mappings, security definitions, listeners, filters, welcome pages, error pages, dependent resources, and many other things. The deployment descriptor for a web application *must* be called web.xml and *must* be placed directly in the WEB-INF subdirectory.

**Design Pattern** A design blueprint for solving a particular problem within a computer application. Design patterns are most commonly associated with the object-oriented community. There is a set of design patterns exclusively associated with J2EE applications.

**DIGEST** A secure form of authentication, which unfortunately doesn't enjoy widespread support on web servers and within browser clients. A digest comprises a highly encrypted form of client credentials, which are useless to any third party who

intercepts them. The challenging server receiving such credentials can, however, compare the encrypted digest with its own database of encrypted digest information, thereby allowing a user access to a protected resource.

**Directive** Directives can be placed within JavaServer Pages to control a number of aspects about the generation of the servlet source representing the JavaServer Page source. A good example is the `<%@ include ...%>` directive, which incorporates the contents of another file within the JavaServer Page at translation time. In their original syntax, directives can be recognized by the `<%@` characters at their beginning. There are now preferred XML equivalent syntaxes for directives used within a JavaServer Page.

**Distributable** A web application can be labeled as distributable by including the `<distributable />` element within the deployment descriptor. This indicates to the application server that the web application need not be confined to one JVM. If the application server supports distributable behavior, it may maintain multiple running copies of the web application across separate JVMs. This is usually for reasons of failure: If one of the web applications fails in one of the JVMs, the other copies will still run. One implication of distributable web applications is that session data for an individual user should be duplicated to all JVMs. The user can then experience continuity of service and no loss of data, even if a JVM terminates unexpectedly.

**Document Type Definition (DTD)** A file used to validate an XML file. The earlier servlet specification (level 2.3) defined DTDs for the deployment descriptor web.xml and tag library descriptor files.

**DTD** *See* **Document Type Definition**.

**EJB** *See* **Enterprise JavaBean**.

**EL** *See* **Expression Language**.

**Element** In XML and HTML, the fundamental building block, potentially comprising a starting tag (with attributes), a body, and a closing tag. For example:

```
<p class="fancy">This is an element</p>
```

**Enterprise JavaBean (EJB)** An Enterprise JavaBean is a J2EE business component. EJBs need an EJB container (typically part of a larger application server) to support their life cycle and other requirements.

**Entity Bean** A kind of EJB that is associated with persistent data—data that doesn't go away even when the application server containing the EJB is shut down. This typically means that an entity bean is associated with a relational database table (though any other form of persistent storage is acceptable as far as the J2EE specification is concerned). An application server provides its own mechanisms for mapping the properties of an entity bean to elements of the persistent storage mechanism (such as tuples in a relational table).

**Error Page** A page defined with the deployment descriptor for a web application. The page can be associated with an HTTP error code (such as 404: page not found) or a Java exception (such as javax.servlet.ServletException). If the defined HTTP error code or Java exception occurs within a request, the corresponding error page is displayed to the user.

***exception*** An implicit object available in a JavaServer Page that is designated as an error page (by setting the isErrorPage of a page directive to a value of true). The exception object is of type java.lang.Throwable and therefore might be used, for example, to obtain stack traces.

**Expression** Within a JavaServer Page, an expression is a piece of Java code that results in direct output to the response.

**Expression Language (EL)** An entire syntax independent of the Java language. EL can be placed in JavaServer Pages to display the output of simple expressions. EL is a preferred alternative to using Java language syntax in normal expressions.

**Extended Markup Language (XML)** A standard for writing text files to contain data in a structured way. Data is held between opening and closing tags. Tags can be nested inside one another in a hierarchical fashion. XML is used for all the key configuration files in a web application, such as the deployment descriptor file and tag library descriptors. XML (or XML-like syntax) is also used for much of JavaServer Page syntax. JSP documents are JavaServer Pages written entirely in XML.

**Filter** An object within a web application that can manipulate a request before it reaches the desired resource, and/or change a response before it is returned to the requester. Typical uses for filters include auditing, logging, transforming, encrypting, de-encrypting, and enforcing security rules.

**Filter Chain** A filter chain is produced when a request hits the outskirts of a web application. The filters that are applicable to the request (as determined by URL mapping) comprise the filter chain, and are ordered according to their order of appearance in the deployment descriptor.

**`<form>`** In HTML, a form can be defined to receive data input from a user. The importance for web applications is that the named fields within the form (defined with HTML tags such as the `<input>` tag) result in request parameters that can be interpreted by servlets and JavaServer Pages.

**FORM** A type of authentication used by web containers as a cosmetic improvement on BASIC authentication. As an alternative to the browser dialog box, a web page is presented for user ID and password entry.

**Forward** If one resource ( JSP or servlet) forwards to another, it is giving responsibility to the other resource for returning a response. Forwarding actions are usually implemented by a RequestDispatcher object.

**Front Controller** A design pattern whereby a component (usually a servlet) acts as a gateway for requests. This component usually takes responsibility for centralized actions, especially controlling navigation between resources.

**GET** An HTTP method that requests the return of a resource from a specified URL.

**HEAD** An HTTP method that requests the header information (metadata) for a resource from a specified URL. HEAD should return everything that is returned by GET, except for the resource itself.

**HTTP Methods** An HTTP method is a mandatory component of an HTTP request. There are seven bona fide HTTP methods that can be associated with HTTP requests: GET, POST, PUT, DELETE, HEAD, OPTIONS, and TRACE.

**HttpSession** Objects implementing the javax.servlet.http.HttpSession interface are created within web containers to overcome the connectionless nature of HTTP. They act as a repository of data that needs to be remembered between requests from an individual user's browser session.

**HyperText Markup Language (HTML)** The language used to construct web pages. Although primarily used for formatting text, HTML defines some tags that control interactivity in the web client. *See* **`<form>`**.

**HyperText Transfer Protocol (HTTP)** A high-level communications protocol situated above Transmission Control Protocol and Internet Protocol (TCP/IP). HTTP is the conduit for most requests on the Internet. The protocol works through requests and responses. Communication between two computers is not meant to be continuous: An interaction typically lasts for one request and one response (it's a connectionless protocol). A whole layer of servlet design (in the javax.servlet.http package) is dedicated to the interpretation and control of HTTP requests and responses. Each HTTP request contains an HTTP method, which can be trapped by a servlet method: GET by `doGet()`, POST by `doPost()`, and so on.

**Idempotent** An action that can be repeated without causing any additional changes. Therefore, "read-only" actions such as an HTTP GET are idempotent. However, idempotent methods are not restricted to "read-only." An HTTP PUT, which places a file on the remote server, is also thought of as idempotent because even when the action is repeated for the same file, the end result will be the same as after the first action. The only nonidempotent method is, in fact, POST. (The idempotent methods are GET, PUT, DELETE, OPTIONS, TRACE, and HEAD.)

**Implicit Objects** In the context of JavaServer Pages, implicit objects are local variables within the `_jspService()` method. Each one represents a significant object in the servlet environment. There are nine in all: *request*, *response*, *out*, *config*, *application*, *session*, *page*, *pageContext* and *exception* (all are defined within this glossary). Implicit objects can be used directly by page authors in, for example, Java scriptlets. Implicit objects are also used liberally by the web container as it generates (nonscriptlet) servlet code for a JavaServer Page.

**Implicit Variables** *See* **Implicit Objects**.

**Include** The process of including another resource within the including resource's response output. Within web applications, this can be achieved in several ways. An include directive will include other resources in a static way, at the point where a JavaServer Page is translated into servlet source. Other forms of include, such as the `include()` method on a RequestDispatcher object and the JavaServer Page `<jsp:include>` custom action, include resources in a dynamic way as each request is run.

**Intercepting Filter** A design pattern that describes an intercepting layer for trapping incoming requests and/or decorating (changing) outgoing responses. This design pattern is built into the servlet specification in a standard way. *See* **Filter**.

**ISO (International Standards Organization)** A body responsible for setting international standards. Character sets for encoding text are described by ISO standards. Of particular relevance to web applications is ISO-8859-1, the default Latin 1 encoding used for HTTP messages.

**Java Archive File (JAR)** A ZIP format file that contains any artifacts needed for a Java application. Within the context of the web application, a JAR file might contain classes that support the activity of other components such as JavaServer Pages.

**Java Message Service** A Java standard API (within J2EE) for communicating in a standard way over messaging middleware (e.g., IBM's MQSeries).

**Java Naming and Directory Interface (JNDI)** A standard within J2EE for locating resources that may be distributed over different JVMs and different computers within a network.

**Java Virtual Machine (JVM)** Software used to run Java code. A JVM transforms Java bytecode (class files) into executable statements for the target platform.

**Java2 Enterprise Edition (J2EE)** A set of Java technologies that augments the standard edition (J2SE). Primarily, J2EE brings together numerous standards and interfaces defining the behavior for enterprise applications written in Java. Some of the more important standards include servlets, JavaServer Pages (JSPs), Enterprise JavaBeans (EJBs), Java Naming and Directory Interface (JNDI), and Java Messaging Service (JMS).

**JavaBean** A simple standard for writing a Java class as an easily reusable component. JavaBeans are especially important for certain JSP custom actions—`<jsp:useBean>`, for example.

**JavaServer Page (JSP)** A text file consisting of native HTML or XML with embedded elements for dynamic content. The embedded elements can consist of script (typically in Java code), XML-like "tags," or a simple expression language (EL). JSPs usually have a .jsp file extension. Once installed in a web container, JavaServer Pages are subject to a "translation phase" that turns the entire JSP source into a Java servlet class.

**JDBC** JDBC describes Java's standard set of APIs for database access. The letters—officially—don't stand for anything.

**JMS** *See* **Java Message Service**.

**JSP** *See* **JavaServer Page**.

**JSP Container** *See* **Container**.

**JSP Document** A form of JavaServer Page source written entirely in XML syntax and often used to output XML files. JSP documents often have a .jspx file extension.

**J2EE** *See* **Java2 Enterprise Edition**.

**JVM** *See* **Java Virtual Machine**.

**Listener** An interface defining methods that can be called by a servlet container when some event occurs. Objects implementing some Listener interface or other can be declared in the web deployment descriptor. The servlet container manages a large range of events. Examples include adding, changing, or removing attributes; starting up or shutting down a web application; and moving a session from one JVM to another in a distributed application.

**Local Variables** Variables declared within a method or received as parameters to a method. These are of particular interest to web component developers

because they are thread safe and are therefore a much better alternative to (say) servlet instance variables. Implicit objects are all defined as local variables in the `_jspService()` method.

**MIME (Multipurpose Internet Mail Extensions)** An Internet standard that defines message formats. Originally developed for email systems, but also embraced by browsers and web servers. The format is very flexible and extensible because MIME messages can embrace many different file types.

**MVC (Model View Controller)** A design pattern, which originated in the Smalltalk community, that separates display logic from the underlying data affecting the display.

**OPTIONS** An HTTP method that returns information about all the HTTP methods allowed for a target resource.

***out*** An implicit object, of type javax.servlet.jsp.JspWriter, used for writing to the response associated with a JavaServer Page.

***page*** An implicit object, of type java.lang.Object, that represents a JavaServer Page. Not typically used directly by JavaServer Page authors.

***pageContext*** An implicit object, of type javax.servlet.jsp.PageContext, that acts as a "master" implicit object for its enclosing JavaServer Page. Most other implicit objects are derived from *pageContext*. Furthermore, *pageContext* is used to store attributes that exist in Page Scope.

**Page Scope** The scope within a JSP from the point where the request is received by that JSP, up to the end of the user request (or until an enclosing JSP or servlet is reached if this comes sooner).

**Pattern** *See* **Design Pattern**.

**POST** An HTTP method that sends data to be processed on a target web server. The results of a POST may be (and often are) of an "updating" nature—perhaps to some underlying database. Consequently, the POST method is not regarded as idempotent, nor is it safe: Users who launch a POST request should be made aware that they may be held accountable. POST has an advantage over GET in that any

associated data is sent in the body of the HTTP request, not as part of the Query String. Consequently, data sent with the POST method can be of any length.

**Programmatic Security** Enforcing security rules through lines of code. J2EE prefers declarative security, but defines APIs to allow the inclusion of programmatic security where required.

**PUT** An HTTP method that places a resource at a target URL. This can be used as a mechanism for uploading files, although most servers disallow it, because any resource already on the server at the location designated by the target URL will be overwritten.

**Query String** A string suffixed to a URL that contains request parameters, and separated by a question mark from the rest of the URL. During execution of an HTTP GET method, form data is appended as request parameters to a query string.

**Relative Path** A path to a resource expressed in partial terms, such as an absolute path, but with information missing from the beginning. Web applications recognize two kinds of relative paths. Relative paths beginning with a forward slash (/) are interpreted as being relative to the context root of the web application. Relative paths without an initial forward slash are interpreted as relative to the location of the resource containing the relative path.

**Relative URI** A partial URL. A relative URI typically misses information from the beginning that would be present in an absolute URL, such as the protocol and host name. *See also* **Relative Path**.

***request*** An implicit object, almost always of type javax.servlet.http.Http ServletRequest, that represents the HTTP request made to a JavaServer Page.

**RequestDispatcher** A RequestDispatcher object can be used to involve other objects (aside from the requested resource) in the production of a response to the user. It can either forward to other resources or include other resources within the current resource's output.

**Request for Comments (RFC)** A series of documents that have evolved into standards for the Internet, maintained by the Internet Engineering Task Force

(IETF: http://www.ietf.org). An RFC may begin as a suggestion and, if popular enough, be adopted as a concrete standard.

**Request Parameter** A piece of information with a name and a value passed within an HTTP request.

**Request Scope** The scope marked out by a single request to a web application—from the request's interception by a web application to its fulfillment.

***response*** An implicit object, almost always of type javax.servlet.http .HttpServletResponse, that represents the HTTP response returned from a JavaServer Page.

**RFC** *See* **Request for Comments**.

**Schema Definition File (XSD File)** An XML file that provides rules about the way another XML document should be constructed. Such an XML file typically has a .xsd extension. In J2EE 2.0, key XML files (such as deployment descriptors and tag library definition files) are now validated against schema definition files. In J2EE 1.2, those same key XML files were validated against document type definition (DTD) files. Schema definition files are often referred to as XSD files.

**Scope** In web applications, the lifetime of variables (attributes) associated with different life cycle objects: page, request, session and context (application).

**Scriptlet** A piece of Java code embedded in a JavaServer Page. The code is placed directly in the `_jspService()` method of the servlet generated from the JavaServer Page source.

**Service Locator** A design pattern that works in conjunction with the Business Delegate design pattern. A Business Delegate component may offload certain responsibilities to a Service Locator—most notably, the code needed to find business services. This is usually (but doesn't have to be) JNDI code.

**Servlet** A Java program that can receive requests and provide responses to a range of clients. Most implementations of servlets provide for HTTP requests, typically originating from a web browser, and return HTTP responses, often in the form of web pages. A servlet needs a servlet container in which to run.

**ServletConfig** A Java object, of type javax.servlet.ServletConfig, created at the point where a servlet is instantiated. Its main role is to give the servlet access to initialization parameters defined in the deployment descriptor.

**Servlet Container** *See* **Container**.

**ServletContext** A Java object, of type javax.servlet.ServletContext, created at the point where a web application is started. Its role is to hold information and give access to APIs that pertain to the web application as a whole.

**Session** A series of interlinked requests.

***session*** An implicit object that contains information about a connected series of requests to a web application (requests are usually deemed to be connected if they originate from the same browser session).

**Session scope** The scope for attributes attached to the same session, thereby permitting a longer duration than the scope of an individual request.

**Simple Tag** An innovation in JSP 2.0 that simplifies the code needed for a tag handler. The number of life cycle events is radically reduced—effectively, responsibility for processing most tag events is handed back to the tag handler developer.

**SSL (Secure Sockets Layer)** A cryptographic protocol designed to provide secure communications on the Internet. It runs beneath protocols such as HTTP, but above the base TCP/IP transport layer.

**Standard Action** *See* **Action**.

**Tag** *See* **Action**.

**Tag File** A JSP-like file that is turned into a simple tag when deployed and used in a JSP container.

**Tag Handler** The Java class file that supports the operation of a custom tag placed in a JSP. Tag handler developers must select from among a number of possible interfaces when writing a tag handler class. The chosen interface determines the

nature and number of life cycle methods that the JSP container calls within the tag handler class. Tag Handlers can be broadly subdivided into classic tag handlers and simple tag handlers.

**Tag Library Descriptor (TLD)** An XML file that defines the characteristics of custom elements that can be placed in JavaServer Pages. Traditionally, this means custom tags, but TLDs can now also be used to define other elements as well, including listeners, EL functions, and tag files.

**taglib Directive** A directive within a JavaServer Page that makes the listeners, EL functions, and custom tags within a tag library descriptor available as actions for use within the page.

**Thread Safety** Servlets are multithreaded by nature. Therefore, within a given servlet container, the same instance of a servlet may have several threads running within its `service()` method simultaneously. This means that local variables are much preferred for use within servlets.

**TRACE** An HTTP method that follows a request from its destination for the purpose of seeing if the request has been changed en route.

**Transfer Object** A design pattern that advises the use of simple JavaBean objects as parameters to and return types from remote Enterprise JavaBean methods. The purpose of this is to reduce and amalgamate the number of remote network calls required to return information from (or pass information to) an EJB object. This pattern was formerly known as the "value object" pattern in previous J2EE versions.

**Translation** In the context of JavaServer pages, the process that turns JavaServer page source into Java servlet source and then compiles this source into a servlet class.

**Translation Unit** An amalgam of a JavaServer Page, any pages statically included in that page, and any pages statically included in those pages (to the end of the static inclusion tree). The entire unit of JavaServer Page source that results is translated into one servlet.

**Uniform Resource Indicator (URI)** The generic name for a means of uniquely identifying a resource within the World Wide Web.

**Uniform Resource Locator (URL)** A string that targets a resource on the Internet, consisting of a protocol (e.g., http), host name (e.g., www.osborne.com), port number (e.g., 80), optional path (e.g., /myfiles), and file name (e.g., index.html, welcome.jsp). A URL is a specific type of URI.

**URL Rewriting** A way of associating all requests having to do with a given session, whereby some unique session ID is appended to all the URLs used in all the requests in the application. Cookies are a preferable way of achieving this aim, but not all client browsers permit the use of cookies.

**Value Object** *See* **Transfer Object**.

**WAR File** *See* **Web Archive**.

**Web Application** In J2EE terms, a web application is a deployable collection of web resources. It comprises many kinds of resources, including servlets, JavaServer Pages, and custom tags. It also supports Java classes, HTML pages, images (GIFs and JPEGs), and a deployment descriptor file. A web application obeys rigid rules about the formation of its directory structure and is often zipped for deployment into a WAR file. *See* **WEB-INF**.

**Web Archive (WAR)** A file in Java archive (zipped) format that contains all the artifacts comprising a web application.

**Web Module** A synonym for web application. Web modules are one of several kinds of J2EE modules (EJB modules are another). Several modules (of the same or different types) may be shipped together as part of the same enterprise application.

**web.xml** *See* **Deployment Descriptor**.

**WEB-INF** A special subdirectory within a web application. It contains artifacts to support the working of the web application: Servlet classes, support classes, and tag libraries should all live within WEB-INF or appropriate subdirectories (/WEB-INF/classes for servlet and support classes and /WEB-INF/lib for components within

JAR files). An application server must not allow direct access to /WEB-INF or its subdirectories.

**Welcome Page** A page defined for a web application that will be displayed when no specific resource in the application is requested (because the requesting URL contains only the context root for the web application).

**XML** *See* **Extended Markup Language**.

**XSD** *See* **Schema Definition File**.