# **Java Technologies for Web Applications**

|  |  |  |
| --- | --- | --- |
|  |  |  |

|  |
| --- |
|  |

You can find web applications everywhere on the Internet. Many of these web applications are used to process online forms, tally voting polls, take orders within online stores, allow users to book a flight for their next business trip, or simply to display the user's login name. Which Java technologies do you need to create a web application and which packages do you need to import? The answer depends on what kind of application you're building and what it does.

This article provides an overview of the Java technologies involved in creating various types of web applications, tells you when you might decide to use them, and provides links to get you started. Future articles will cover how to use these technologies in building web applications. To use the technologies described in this article, download [Java Development Kit (JDK) 5.0 Update 9 with Java Platform, Enterprise Edition (Java EE)](http://www.oracle.com/technetwork/java/javase/downloads/index-jsp-138363.html) or later and install it on your computer.

Before you begin coding, you need to understand what a web application is, what Java technologies are available for use, and what development tools can save you time and make creating applications easier.

**Contents**

[What Is a Web Application?](http://www.oracle.com/technetwork/articles/javase/webapps-1-138794.html#webapp)

[Java Technologies to Use in Web Applications](http://www.oracle.com/technetwork/articles/javase/webapps-1-138794.html#technologies)

[Other Technologies to Consider](http://www.oracle.com/technetwork/articles/javase/webapps-1-138794.html#other)

[Development Tools](http://www.oracle.com/technetwork/articles/javase/webapps-1-138794.html#tools)

**What Is a Web Application?**

|  |
| --- |
|  |

Web applications are by nature distributed applications, meaning that they are programs that run on more than one computer and communicate through a network or server. Specifically, web applications are accessed with a web browser and are popular because of the ease of using the browser as a user client. For the enterprise, the ability to update and maintain web applications without deploying and installing software on potentially thousands of client computers is a key reason for their popularity. Web applications are used for web mail, online retail sales, discussion boards, weblogs, online banking, and more. One web application can be accessed and used by millions of people.

Like desktop applications, web applications are made up of many parts and often contain miniprograms, some of which have user interfaces, and some of which do not require a graphical user interface (GUI) at all. In addition, web applications frequently require an additional markup or scripting language, such as HTML, CSS, or JavaScript programming language. Also, many applications use only the Java programming language, which is ideal because of its versatility.

A web application can be as simple as a page that shows the current date and time or as complex as a set of pages on which you can look up and book the most convenient flight, hotels, and car rentals for your next vacation.

The Java technologies you'll use to create web applications are a part of the Java EE platform, in addition to many of the Java Platform, Standard Edition (Java SE) classes and packages. In order for many of these technologies to work on a server, the server must have a container, or web server, installed that recognizes and runs the classes you create. For development and testing of these technologies, you can use the tools detailed in this article, but when you deploy, make sure that the server has Java server software installed to run Java technology-based web applications. If you don't have access to this information, ask the server administrator.

**Java Technologies to Use in Web Applications**

|  |
| --- |
|  |

There are too many Java technologies to list in one article, so this article will describe only the ones most frequently used. The number of technologies listed here can appear overwhelming. Keep in mind that you will not need to use them all. In fact, a web application often consists of nothing more than one page created with the JavaServer Pages (JSP) technology. Sometimes you will combine three or more such technologies. No matter how many you end up using, it's good to know what is available to you and how you can use each one in a web application.

**Java Servlet API**

The [Java Servlet API](http://www.oracle.com/technetwork/java/whitepaper-135196.html) lets you define HTTP-specific classes. A servlet class extends the capabilities of servers that host applications that are accessed by way of a request-response programming model. Although servlets can respond to any type of request, they are commonly used to extend the applications hosted by web servers. For instance, you might use a servlet to get the text input from an online form and print it back to the screen in an HTML page and format, or you might use a different servlet to write the data to a file or database instead. A servlet runs on the server side -- without an application GUI or HTML user interface (UI) of its own. Java Servlet extensions make many web applications possible.

Figure 1 shows clients talking to Java Servlet extensions. Clients may range in complexity from simple HTML forms to sophisticated Java technology-based applets.

The [javax.servlet](http://www.oracle.com/technetwork/java/overview-137084.html) and [javax.servlet.http](http://www.oracle.com/technetwork/java/overview-137084.html) packages provide the classes and interfaces to define servlets. HTML servlet classes extend the javax.servlet.http.HttpServlet abstract class, which provides a framework for handling HTTP protocol.

Learn more about [Java Servlet extensions](http://www.oracle.com/technetwork/java/index-jsp-135475.html) and [navigating with servlets](http://www.oracle.com/technetwork/topics/newtojava/servlets-142927.html).

**JavaServer Pages Technology**

JavaServer Pages (JSP) technology provides a simplified, fast way to create dynamic web content. JSP technology enables rapid development of web-based applications that are server- and platform-independent. JSP technology lets you add snippets of servlet code directly into a text-based document. Typically, a JSP page is a text-based document that contains two types of text:

* Static data, which can be expressed in any text-based format, such as HTML, Wireless Markup Language (WML), or XML
* JSP technology elements, which determine how the page constructs dynamic content

The packages involved in creating JSP pages are javax.el, javax.servlet.jsp, javax.servlet.jsp.el, and javax.servlet.jsp.tagext, though you will rarely have to import these directly. A JSP page can be as simple as a bit of HTML with one snippet of JSP code and the .jsp extension of the page name.

For instance, you can create a web site of JSP technology pages that use one snippet of code to include the header.html file, which contains the site navigation. This way, when you change a link to a button in the navigation, you make the change in only one file, and that file loads into all the pages on the site that have this code snippet:

<%@ include file="header.html" %>

That line of code works very much like a server-side include, if you are familiar with those. Because this web page is now a JSP page, you could also go on to add more Java technology code to create dynamic web content, such as polls, forms, ways to enter or retrieve data from a database, and so forth.

Take a look at two helpful documents about creating JSP pages:

* [JavaServer Pages (JSP) v2.0 Syntax Reference](http://www.oracle.com/technetwork/java/syntaxref20-136787.html)
* [JavaServer Pages Technology -- Frequently Asked Questions](http://www.oracle.com/technetwork/java/faq-137059.html)

**JavaServer Pages Standard Tag Library**

The JavaServer Pages Standard Tag Library (JSTL) encapsulates core functionality common to many JSP technology-based applications. Instead of mixing tags from numerous vendors in your applications, you employ a single standard set of tags. This standardization allows you to deploy your applications on any JSP container that supports JSTL and makes it more likely that the implementation of the tags is optimized.

JSTL has iterator and conditional tags for handling flow control, tags for manipulating XML documents, internationalization tags, tags for accessing databases using SQL, and tags for commonly used functions.

The packages you can access for using JSTL are [javax.servlet.jsp.jstl.core](http://docs.oracle.com/javaee/5/api/javax/servlet/jsp/jstl/core/package-summary.html) , [javax.servlet.jsp.jstl.fmt](https://glassfish.dev.java.net/nonav/javaee5/api/javax/servlet/jsp/jstl/fmt/package-summary.html) , [javax.servlet.jsp.jstl.sql](https://glassfish.dev.java.net/nonav/javaee5/api/javax/servlet/jsp/jstl/sql/package-tree.html) , and [javax.servlet.jsp.jstl.tlv](http://docs.oracle.com/cd/E17802_01/webservices/webservices/docs/1.5/api/javax/servlet/jsp/jstl/tlv/package-summary.html) .

Learn more about [JSTL](http://www.oracle.com/technetwork/java/jstl-137486.html).

**JavaServer Faces Technology**

JavaServer Faces technology is a UI framework for building web applications. The main components of JavaServer Faces technology involve a GUI component framework, a flexible model for rendering components in various markup languages and technologies, and a standard RenderKit for generating HTML markup.

This functionality is available through standard Java APIs and XML-based configuration files. In addition, [Sun Java Studio Creator IDE](http://www.oracle.com/technetwork/articles/java/webapps-1-138794.html#) leverages JavaServer Faces technology in its drag-and-drop GUI tools, allowing you to use the technology without having to write or understand the underlying code. See also " [Getting Started With Sun Java Studio Creator](http://www.oracle.com/technetwork/articles/java/webapps-1-138794.html#)."

**Java Message Service API**

*Messaging* is a method of communication between software components or applications. A messaging system is a peer-to-peer facility. In other words, a messaging client can send messages to and receive messages from any other client. Each client connects to a messaging agent that provides facilities for creating, sending, receiving, and reading messages. By combining Java technology with enterprise messaging, the Java Message Service (JMS) API provides a powerful tool for solving enterprise computing problems.

Enterprise messaging provides a reliable, flexible service for the exchange of business data throughout an enterprise. The JMS API adds to this a common API and provider framework that enables the development of portable message-based applications in the Java programming language. An example of how JMS might be used is an application that keeps track of inventory for an auto manufacturer.The inventory component can send a message to the factory component when the inventory level for a product goes below a certain level, so the factory can make more cars. The factory component can send a message to the parts components so that the factory can assemble the parts it needs.The parts components in turn can send messages to their own inventory and order components to update their inventories and to order new parts from suppliers and so forth.

The JMS API improves programmer productivity by defining a common set of messaging concepts and programming strategies that all JMS technology-compliant messaging systems will support.

Learn more about the [JMS API](http://www.oracle.com/technetwork/java/jms-136181.html).

**JavaMail API and the JavaBeans Activation Framework**

Web applications can use the JavaMail API to send email notifications. The API has two parts: an application-level interface that the application components use to send email and a service provider interface. Service providers implement particular email protocols, such as SMTP. Several service providers are included with the JavaMail API package, and others are available separately. The Java EE platform includes the JavaMail extension with a service provider that allows application components to send email.

In conjunction with the JavaMail extension, you might use the JavaBeans Activation Framework (JAF) API. This API provides standard services to determine the type of an arbitrary piece of data, encapsulate access to it, discover the operations available on it, and create the appropriate component based on JavaBeans component architecture (JavaBeans component) to perform those operations.

The JavaMail API is now open source. See the [GlassFish Project -- JavaMail](https://glassfish.dev.java.net/javaee5/mail/) home page.

Learn more about the [JavaMail API](http://www.oracle.com/technetwork/java/javamail/javamail-138606.html) and the [JAF](http://www.oracle.com/technetwork/java/javase/jaf-135115.html).

**Java API for XML Processing**

The Java API for XML Processing (JAXP), part of the Java SE platform, supports the processing of XML documents using the Document Object Model (DOM), the Simple API for XML (SAX), and Extensible Stylesheet Language Transformations (XSLT). JAXP enables applications to parse and transform XML documents independent of a particular XML-processing implementation.

JAXP also provides namespace support, which lets you work with schemas that might otherwise have naming conflicts. Designed to be flexible, JAXP lets you use any XML-compliant parser or XSL processor from within your application and supports the W3C schema.

Learn more about [JAXP](https://jaxp.dev.java.net/).

The Java API for XML Web Services (JAX-WS) is now an [open-source project](https://jax-ws.dev.java.net) at java.net and is also a key part of the [GlassFish](https://glassfish.dev.java.net) project. See also [Introducing JAX-WS 2.0 With the Java SE 6 Platform, Part 1](http://www.oracle.com/technetwork/articles/javase/jax-ws-2-141894.html).

**JDBC API**

The JDBC API allows you invoke database SQL commands from Java programming language methods. You can use the JDBC API in a servlet, JSP technology page, or an enterprise bean when you need to access the database.

The JDBC API has two parts: an application-level interface that application components use to access a database and a service provider interface to attach a JDBC driver to the Java EE platform.

The left side of Figure 2 shows how a client makes a direct call to the application server, usually through a servlet or JSP page, and that data is then sent to the database management system (DBMS) server. The right side of Figure 2 shows how a driver translates JDBC calls into the middleware vendor's protocol.

Learn more about the [JDBC API](http://docs.oracle.com/javase/1.5.0/docs/guide/jdbc/index.html).

**Java Persistence API**

The Java Persistence API is a Java technology standards-based solution for persistence. Persistence uses an object-relational mapping approach to bridge the gap between an object-oriented model and a relational database. Java technology persistence consists of three areas:

* The Java Persistence API
* The query language
* Object-relational mapping metadata

Read the [Java Persistence API FAQ](http://www.oracle.com/technetwork/java/javaee/persistence-jsp-136066.html).

**Java Naming and Directory Interface**

The Java Naming and Directory Interface (JNDI) provides naming and directory functionality, enabling applications to access multiple naming and directory services. It provides applications with methods for performing standard directory operations, such as associating attributes with objects and searching for objects using their attributes. Using JNDI, a web application can store and retrieve any type of named Java technology object, allowing applications to coexist with many legacy applications and systems.

Naming services provide application clients, enterprise beans, and web components with access to a JNDI naming environment. A naming environment allows the developer to customize a component without having to access or change the component's source code. A container implements the component's environment and provides it to the component as a JNDI naming context.

Learn more about [JNDI](http://www.oracle.com/technetwork/java/jndi-136720.html).

**Other Technologies to Consider**

|  |
| --- |
|  |

You might want to consider using the following other technologies in your web application, depending on the application's complexity:

* [J2EE Connector Architecture](http://www.oracle.com/technetwork/java/javaee/overview/connector-jsp-135375.html): Tools vendors and system integrators use this architecture to create resource adapters that support access to enterprise information systems that can be plugged in to any Java EE technology-based product.
* [Java Authentication and Authorization Service (JAAS)](http://www.oracle.com/technetwork/articles/java/ssLINK/137016): JAAS provides a way for a Java EE technology-based application to authenticate and authorize a specific user or group of users to run it.
* [Java API for XML Registries (JAXR)](http://www.oracle.com/technetwork/java/index-jsp-140032.html): JAXR lets you access business and general-purpose registries over the web.
* [Java Architecture for XML Binding (JAXB)](http://jaxb.java.net/): JAXB provides a convenient way to bind an XML schema to a representation in applications written in the Java programming language.
* [SOAP with Attachments API for Java (SAAJ)](http://www.oracle.com/technetwork/java/faq-142237.html): SAAJ is a low-level API on which JAX-WS and JAXR depend.
* [Java Transaction API (JTA)](http://www.oracle.com/technetwork/java/remove-jsp-135469.html#1): JTA provides a standard interface for demarcating transactions.

**Development Tools**

|  |
| --- |
|  |

Integrated development environments (IDEs) can speed the process of developing and testing web applications. It's no wonder that developers move to an IDE, but which one is right for you? Which features do you need? Sun Microsystems supports three IDEs for the Java platform: NetBeans IDE, Sun Java Studio Creator IDE, and Sun Java Studio Enterprise IDE.

**NetBeans IDE**

NetBeans IDE is free and open source. This IDE is written in the Java programming language and provides the services common to desktop applications, such as window and menu management, settings storage, and so forth. It is also the first IDE to fully support JDK 5.0 features. In addition, it supports web tools, including the NetBeans IDE Enterprise Pack, which adds everything you need to immediately start writing, testing, and debugging Java technology-based web applications.

The pack provides visual design tools for Unified Modeling Language (UML) modeling, XML schema creation, modification, and visualization, as well as development of secure, identity-enabled web services.

**Sun Java Studio Creator IDE**

[Sun Java Studio Creator IDE](http://www.oracle.com/technetwork/articles/java/webapps-1-138794.html#) is great for quick and easy web application development. In addition, this IDE is built on the NetBeans IDE, starting with a subset of the functionality and extending it.

Sun Java Studio Creator IDE allows you to build applications visually. The programming part is cleanly separated from the UI. With Sun Java Studio Creator IDE, the IDE takes care of a lot of the repetitive coding behind the UI.

The rapid visual drag-and-drop features are based on JavaServer Faces technology, which is a framework for building UIs for web applications. A GUI is used for manipulating JavaServer Faces components, as well as visually defining page flow. There's also graphical support for easily working with databases and web services.

**Sun Java Studio Enterprise IDE**

[Sun Java Studio Enterprise IDE](http://www.oracle.com/technetwork/articles/java/webapps-1-138794.html#) is a powerful set of tools that provides an integrated framework for enterprise-grade, rapid web application development. It offers enhanced debugging and development support for web services and for development of Java EE technology-based applications.

Sun Java Studio Enterprise IDE is built on and extends the NetBeans IDE. It also enables smart code editing, such as completing your code for you, with refactoring, renaming objects throughout an application. It also enables performance tuning -- ensuring better use of memory and so forth -- of applications with an optimal end- user experience.

Sun Java Studio Enterprise IDE provides a model-driven analysis, design, and development environment that leverages UML. This integrated feature reduces complexity and increases visual clarity across software development projects, ensuring that a sound architecture is established and communicated throughout the enterprise.

Now that you are familiar with the many Java technologies that you use in web applications, you are ready to develop your own application. In an upcoming series of articles, you will learn to create web applications using Sun Java Studio Creator IDE, which you can download for free, and learn how to use the Java technologies mentioned in this article. You will first create small simple applications, then work up to more complex programs. In the meantime, follow the links in this article to learn more about the Java technologies discussed in this article.

**For More Information**

|  |
| --- |
|  |

* [Java Message Service (JMS) API](http://www.oracle.com/technetwork/java/jms-136181.html)
* [JDBC API](http://docs.oracle.com/javase/1.5.0/docs/guide/jdbc/index.html)
* [Java Persistence API FAQ](http://www.oracle.com/technetwork/java/javaee/persistence-jsp-136066.html)
* [Getting Started with an Integrated Development Environment (IDE)](http://www.oracle.com/technetwork/articles/java/ssLINK/141450)
* [JavaMail API](http://www.oracle.com/technetwork/java/javamail/javamail-138606.html)
* [Java Web Services Developer Pack (Synopsis of Components)](http://www.oracle.com/technetwork/java/faq-142237.html)