**EJB**

# **What is EJB**

EJB is an acronym for *enterprise java bean*. It is a specification provided by Sun Microsystems to develop secured, robust and scalable distributed applications.

To run EJB application, you need an *application server* (EJB Container) such as Jboss, Glassfish, Weblogic, Websphere etc. It performs:

1. life cycle management,
2. security,
3. transaction management, and
4. object pooling.

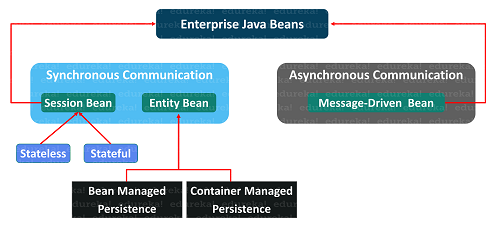
EJB application is deployed on the server, so it is called server side component also.

EJB is like COM (*Component Object Model*) provided by Microsoft. But, it is different from Java Bean, RMI and Web Services.

## **When use Enterprise Java Bean?**

1. **Application needs Remote Access**. In other words, it is distributed.
2. **Application needs to be scalable**. EJB applications supports load balancing, clustering and fail-over.
3. **Application needs encapsulated business logic**. EJB application is separated from presentation and persistent layer.

## **Types of Enterprise Java Bean**



There are 3 types of enterprise bean in java.

#### Session Bean

Session bean contains business logic that can be invoked by local, remote or webservice client.

There are two types of session beans:

* [Stateless Session Beans](https://docs.oracle.com/cd/A97688_16/generic.903/a97677/overview.htm" \l "1006783)--Stateless session beans do not share state or identity between method invocations. They are useful mainly in middle-tier application servers that provide a pool of beans to process frequent and brief requests.

* [Stateful Session Beans](https://docs.oracle.com/cd/A97688_16/generic.903/a97677/overview.htm" \l "1006395)--Stateful session beans are useful for conversational sessions, in which it is necessary to maintain state, such as instance variable values or transactional state, between method invocations. These session beans are mapped to a single client for the life of that client.

#### Message Driven Bean

#### 

#### 

#### Message-driven beans are enterprise beans that receive and process Java message service messages. They can be accessed only through messaging. They do not have interfaces. Asynchronous communication between the queue and the listener takes place.

#### Entity Bean

It encapsulates the state that can be persisted in the database. It is deprecated. Now, it is replaced with JPA (Java Persistent API).

These beans contain persistent data and it can be saved in the data source. There are two types:

* Container managed persistence: these entity beans assign their persistence to the EJB container
* Bean managed persistence: these entity beans manage their own persistence.

## **Disadvantages of EJB**

1. Requires application server
2. Requires only java client. For other language client, you need to go for webservice.
3. Complex to understand and develop ejb applications.

**Struts**

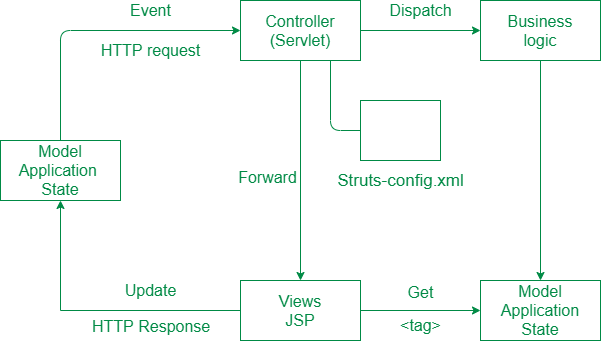
**Struts** is used to create a **web applications based on servlet and JSP**. Struts depend on the MVC (Model View Controller) framework. Struts application is a genuine web application. Struts are thoroughly useful in building J2EE (Java 2 Platform, Enterprise Edition) applications because struts takes advantage of J2EE design patterns. Struts follows these J2EE design patterns including MVC.

In struts, the composite view manages the layout of its sub-views and can implement a template, making persistent look and feel easier to achieve and customize across the entire application. A composite view is made up by using other reusable sub views such that a small change happens in a sub-view is automatically updated in every composite view.

Struts consists of a set of own custom tag libraries. Struts are based on MVC framework which is pattern oriented and includes JSP custom tag libraries. Struts also supports utility classes.

**Features of Struts:** Struts has the following features:

* Struts encourages good design practices and modeling because the framework is designed with “time-proven” design patterns.
* Struts is almost simple, so easy to learn and use.
* It supports many convenient features such as input validation and internationalization.
* It takes much of the complexity out as instead of building your own MVC framework, you can use struts.
* Struts is very well integrated with J2EE.
* Struts has large user community.
* It is flexible and extensible, it is easy for the existing web applications to adapt the struts framework.
* Struts provide good tag libraries.
* It allows capturing input form data into javabean objects called Action forms.
* It also hand over standard error handling both programmatically and declaratively.

**Working of Struts:**  


In the initialization phase, the controller rectify a configuration file and used it to deploy other control layer objects. Struts configuration is form by these objects combined together. The struts configuration defines among other things the action mappings for an application.  
Struts controller servlet considers the action mappings and routes the HTTP requests to other components in the framework. Request is first delivered to an action and then to JSP. The mapping helps the controller to change HTTP requests into application actions. The action objects can handle the request from and responds to the client (generally a web browser). Action objects have access to the applications controller servlet and also access to the servlet’s methods. When delivering the control, an action objects can indirectly forward one or more share objects, including javabeans by establish them in the typical situation shared by java servlets.

**Hibernate**

Hibernate is an **O**bject-**R**elational **M**apping (ORM) solution for JAVA. It is an open source persistent framework created by Gavin King in 2001. It is a powerful, high performance Object-Relational Persistence and Query service for any Java Application.

Hibernate maps Java classes to database tables and from Java data types to SQL data types and relieves the developer from 95% of common data persistence related programming tasks.

Hibernate sits between traditional Java objects and database server to handle all the works in persisting those objects based on the appropriate O/R mechanisms and patterns.



Hibernate Advantages

* Hibernate takes care of mapping Java classes to database tables using XML files and without writing any line of code.
* Provides simple APIs for storing and retrieving Java objects directly to and from the database.
* If there is change in the database or in any table, then you need to change the XML file properties only.
* Abstracts away the unfamiliar SQL types and provides a way to work around familiar Java Objects.
* Hibernate does not require an application server to operate.
* Manipulates Complex associations of objects of your database.
* Minimizes database access with smart fetching strategies.
* Provides simple querying of data.

Supported Databases

Hibernate supports almost all the major RDBMS. Following is a list of few of the database engines supported by Hibernate −

* HSQL Database Engine
* DB2/NT
* MySQL
* PostgreSQL
* FrontBase
* Oracle
* Microsoft SQL Server Database
* Sybase SQL Server
* Informix Dynamic Server

Supported Technologies

Hibernate supports a variety of other technologies, including −

* XDoclet Spring
* J2EE
* Eclipse plug-ins
* Maven

Spring

Spring framework is an open source Java platform that provides comprehensive infrastructure support for developing robust Java applications very easily and very rapidly. Spring framework was initially written by Rod Johnson and was first released under the Apache 2.0 license in June 2003. The core features of the Spring Framework can be used in developing any Java application, but there are extensions for building web applications on top of the Java EE platform. Spring framework targets to make J2EE development easier to use and promotes good programming practices by enabling a POJO-based programming model.

## **Advantages /Applications of Spring**

Following is the list of few of the great benefits of using Spring Framework −

* **POJO Based** - Spring enables developers to develop enterprise-class applications using POJOs. The benefit of using only POJOs is that you do not need an EJB container product such as an application server but you have the option of using only a robust servlet container such as Tomcat or some commercial product.
* **Modular** - Spring is organized in a modular fashion. Even though the number of packages and classes are substantial, you have to worry only about the ones you need and ignore the rest.
* **Integration with existing frameworks** - Spring does not reinvent the wheel, instead it truly makes use of some of the existing technologies like several ORM frameworks, logging frameworks, JEE, Quartz and JDK timers, and other view technologies.
* **Testablity** - Testing an application written with Spring is simple because environment-dependent code is moved into this framework. Furthermore, by using JavaBeanstyle POJOs, it becomes easier to use dependency injection for injecting test data.
* **Web MVC** - Spring's web framework is a well-designed web MVC framework, which provides a great alternative to web frameworks such as Struts or other over-engineered or less popular web frameworks.
* **Central Exception Handling** - Spring provides a convenient API to translate technology-specific exceptions (thrown by JDBC, Hibernate, or JDO, for example) into consistent, unchecked exceptions.
* **Lightweight** - Lightweight IoC containers tend to be lightweight, especially when compared to EJB containers, for example. This is beneficial for developing and deploying applications on computers with limited memory and CPU resources.
* **Transaction management** - Spring provides a consistent transaction management interface that can scale down to a local transaction (using a single database, for example) and scale up to global transactions (using JTA, for example).

## Architecture

Spring is well-organized architecture consisting  of seven modules. Modules in the Spring framework are:

### Spring AOP

One of the key components of Spring is the *AOP framework*. AOP is used in Spring:

* + To provide declarative enterprise services, especially as a replacement for EJB declarative services. The most important such service is *declarative transaction management*, which builds on Spring’s transaction abstraction.
  + To allow users to implement custom aspects, complementing their use of OOP with AOP

### Spring ORM

The *ORM* package is related to the database access. It provides integration layers for popular object-relational mapping APIs, including JDO, Hibernate and iBatis.

### Spring Web

The Spring Web module is part of Spring?s web application development stack, which includes Spring MVC.

### Spring DAO

The DAO (Data Access Object) support in Spring is primarily for standardizing the data access work using the technologies like JDBC, Hibernate or JDO.

### Spring Context

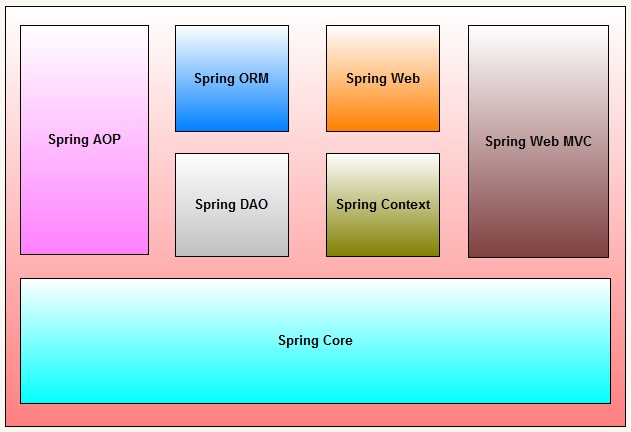
This package builds on the beans package to add support for message sources and for the Observer design pattern, and the ability for application objects to obtain resources using a consistent API.

### Spring Web MVC

This is the Module which provides the MVC implementations for the web applications.

### Spring Core

The *Core* package is the most import component of the Spring Framework.  
This component provides the Dependency Injection features. The BeanFactory  provides a factory pattern which separates the dependencies like initialization, creation and access of the objects from your actual program logic.



**JSP**

**JSP** technology is used to create web application just like Servlet technology. It can be thought of as an extension to Servlet because it provides more functionality than servlet such as expression language, JSTL, etc.

A JSP page consists of HTML tags and JSP tags. The JSP pages are easier to maintain than Servlet because we can separate designing and development. It provides some additional features such as Expression Language, Custom Tags, etc.

Advantages of JSP over Servlet

There are many advantages of JSP over the Servlet. They are as follows:

1) Extension to Servlet

JSP technology is the extension to Servlet technology. We can use all the features of the Servlet in JSP. In addition to, we can use implicit objects, predefined tags, expression language and Custom tags in JSP, that makes JSP development easy.

#### 2) Easy to maintain

JSP can be easily managed because we can easily separate our business logic with presentation logic. In Servlet technology, we mix our business logic with the presentation logic.

#### 3) Fast Development: No need to recompile and redeploy

If JSP page is modified, we don't need to recompile and redeploy the project. The Servlet code needs to be updated and recompiled if we have to change the look and feel of the application.

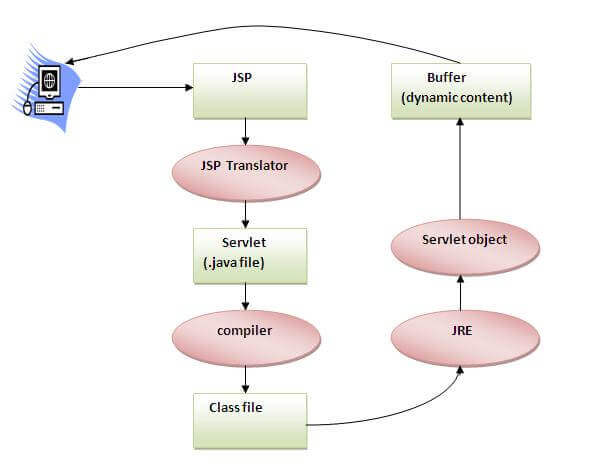
#### 4) Less code than Servlet

In JSP, we can use many tags such as action tags, JSTL, custom tags, etc. that reduces the code. Moreover, we can use EL, implicit objects, etc.

The Lifecycle of a JSP Page

The JSP pages follow these phases:

* Translation of JSP Page
* Compilation of JSP Page
* Classloading (the classloader loads class file)
* Instantiation (Object of the Generated Servlet is created).
* Initialization ( the container invokes jspInit() method).
* Request processing ( the container invokes \_jspService() method).
* Destroy ( the container invokes jspDestroy() method).



As depicted in the above diagram, JSP page is translated into Servlet by the help of JSP translator. The JSP translator is a part of the web server which is responsible for translating the JSP page into Servlet. After that, Servlet page is compiled by the compiler and gets converted into the class file. Moreover, all the processes that happen in Servlet are performed on JSP later like initialization, committing response to the browser and destroy.

Creating a simple JSP Page

To create the first JSP page, write some HTML code as given below, and save it by .jsp extension. We have saved this file as index.jsp. Put it in a folder and paste the folder in the web-apps directory in apache tomcat to run the JSP page.

**index.jsp**

Let's see the simple example of JSP where we are using the scriptlet tag to put Java code in the JSP page. We will learn scriptlet tag later.

1. <html>
2. <body>
3. <% out.print(2\*5); %>
4. </body>
5. </html>

It will print **10** on the browser.

How to run a simple JSP Page?

Follow the following steps to execute this JSP page:

* Start the server
* Put the JSP file in a folder and deploy on the server
* Visit the browser by the URL http://localhost:portno/contextRoot/jspfile, for example, <http://localhost:8888/myapplication/index.jsp>

# **Servlet vs JSP: What's the Difference?**

## What Is Servlet?

A servlet is a Java technology that is managed by a container called a servlet engine. It generates dynamic content and interacts with the client through Request and Response. Servlet extends the functionality of a web server.

Though servlets can respond to many types of requests, they generally implement web containers for hosting any website on web server. Therefore, it is qualified as a server-side servlet web API.

## What Is JSP?

JSP is a collection of technologies developed by Sun Microsystems. It is used to develop web pages by inserting Java code into the HTML pages by making special JSP tags. It can consist of either HTML or XML (combination of both is also possible) with JSP actions and commands. The full form of JSP is Java Server Pages.

Top 10 Behavioral Interview Questions and Answers

## KEY DIFFERENCES

* Servlet can accept all protocol requests, including HTTP, while JSP can only accept HTTP requests.
* In MVC architecture, servlet works as a controller while JSP works as a view for displaying output.
* Servlet should be used when there is more data processing involved whereas, JSP is generally used when there is less involvement of data processing.
* Servlets run faster than JSP, on the other hand JSP runs slower than servlet as it takes time to compile the program and convert into servlets.
* You can override the service() method in servlet but, in JSP, you can't override the service() method.
* In Servlet, you have to implement both business logic and presentation logic in the single file. Whereas in JSP, business logic is split from presentation logic using JavaBeans.

## Why use Servlet?

Here, are the reasons of using Servlet

* The performance is much better.
* Servlet is used when you do not like to create a separate process to handle each and every request of client.
* Developers can use servlets when they have to take advantage of all the features of Java.
* Java class libraries which are available to a servlet can communicate with applets, databases, or other software via RMI and sockets mechanisms.

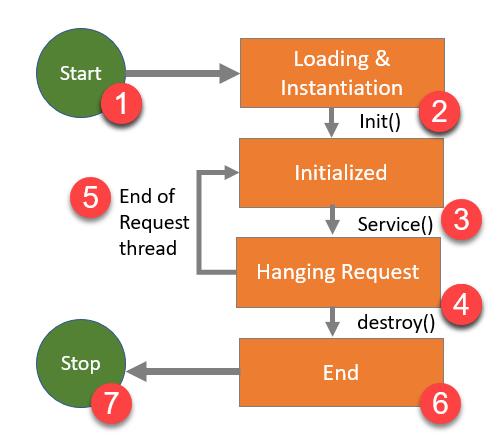
## Why use JSP?

Here, are reasons of using JSP:

* In Java server pages JSP, the execution is much faster compared to other dynamic languages.
* It is much better than Common Gateway Interface (CGI).
* Java server pages (JSP)are always compiled before its processed by the server as it reduces the effort of the server to create process.
* Java server pages are built over Servlets API. Hence, it has access to all Java APIs, JNDI, JDBC EJB, and other components of java.
* JSP is an important part of Java EE (Enterprise Edition), which is a platform for enterprise-level applications.

## Servlet Life Cycle

Servlet life cycle is depicted in the below diagram:

[](https://cdn.guru99.com/images/2/022220_0728_ServletvsJS1.png)Servlet Life Cycle

**1) Start:**Execution of servlet begins.

**2) Loading & instantiation void init():** It is called when servlet is first loaded. This method lets you initialize servlet.

**3)** **Initialized** **void service()**: The purpose of this method is to serve a request. You can call it as many times as you like.

**4) Handling request and destroying servlet:** Java application must be first determined what code is needed to execute the request URL to provide a response. To destroy servlet Void destroy method is used at the end of servlet life cycle.

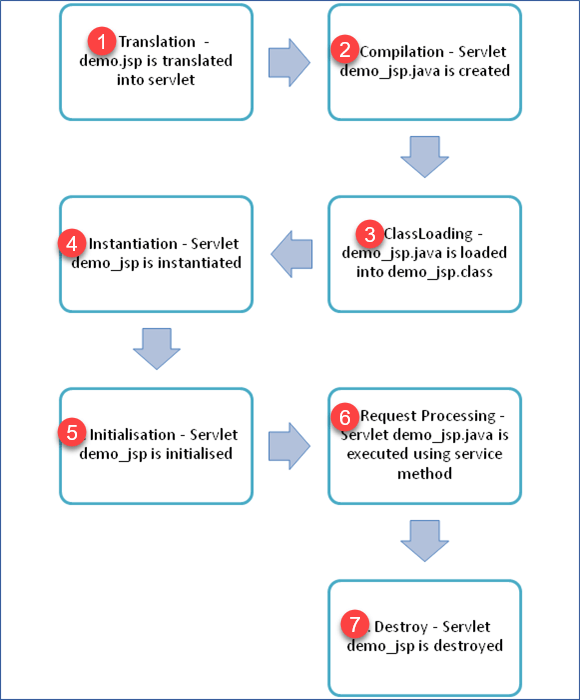
**5) End of Request Thread:**When service() finishes its task, either the thread ends or returns to the thread pool that is managed by servlet contaier.

**6) End:**Servlet lifecycle finishes.

**7): Stop:** Servlet stop executing.

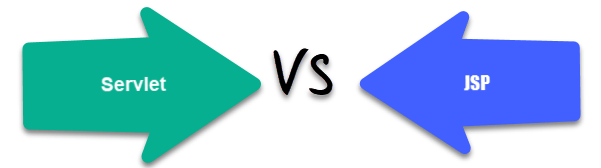
## JSP Life Cycle

JSP Lifecycle is depicted in the diagram below.

[](https://cdn.guru99.com/images/2/022220_0728_ServletvsJS2.png)JSP Life Cycle

1. Translation of JSP page into a servlet.
2. Compilation of JSP page(Compilation of JSP page into \_jsp.java)
3. Classloading (\_jsp.java is converted to class file \_jsp.class)
4. Instantiation(Object of generated servlet is created)
5. Initialisation(\_jspinit() method is invoked by container)
6. Request Processing(\_jspservice() method is invoked by the container)
7. Destroy (\_jspDestroy() method invoked by the container)

## Difference between a Servlet and JSP

[](https://cdn.guru99.com/images/2/022220_0728_ServletvsJS3.png)

Here are the main differences between Servlet and JSP:

|  |  |
| --- | --- |
| **Servlet** | **JSP** |
| Servlets run faster than JSP. | JSP runs slower than servlet as it takes time to compile the program and convert into servlets. |
| It is hard to write code in servlet. | It's easier to code in JSP compared to servlets. |
| In MVC architecture, servlet works as a controller. | In MVC architecture, JSP works as a view for displaying output. |
| It should be use when there is more data processing involved. | JSP is generally used when there is no involvement of much data processing. |
| There is no custom tag writing facility in servlets. | You can easily build custom tags that can directly call Java beans. |
| Servlet is a java code. | JSP is a HTML-based code. |
| It can accept all protocol requests, including HTTP. | It can only accept HTTP requests. |
| You can override the service() method. | In JSP, you can't override the service() method. |
| In Servlet, by default, session management is not enabled, user has to enable it explicitly. | In JSP, session management is automatically enabled. |
| In Servlet, you have to implement both business logic and presentation logic in the single file. | In JSP, business logic is split from presentation logic using JavaBeans. |
| Modification in Servlet file is a time consuming due to reloading, recompiling, and restarting the server. | JSP modification is fast, as you just need to click one refresh button. |

## Advantages of Servlet

Here are benefits/ pros of using servlet:

* Servlet loads only one copy of it into JVM (Java Virtual Machine).
* It saves time to respond to the first request which increases response time.
* It uses standard API that is supported by numerous web servers.
* Programming languages like platform independence and ease of development.
* You can access the large set of APIs that are available for the Java platform.
* The web container makes threads for handling more than one request to the servlet.
* Easy coordination between two or more servlet to make web applications.
* Servlet containers support many other features like sessions, resource management, persistence, security, etc.
* Servlet usage doesn't constrain by the web browser.

## Disadvantages of Servlet

Here are cons/drawbacks for using servlet:

* One servlet is loaded into JVM. It does matter numbers of requests.
* When there is a request, there is a thread, not a process.
* Servlet is persistent until it destroys.
* Designing in a servlet is difficult and slows down the application.
* You need a JRE(Java Runtime Environment) on the server to run servlets.
* For non-java developers, servlet is not suitable as they required to have a broad knowledge of Java servlet.
* HTML code is mixed up with Java code therefore, changes done in one code can affect another code.
* Writing HTML code in servlet programming is very difficult. It also makes servlet looks bulky.
* In servlet programming, if you want to use implicit objects, you need to write some additional code in order to access them.
* Developers must take care of exception handling because servlet programming is not thread-safe by default.

## Advantages of JSP

Here, are benefits/ pros of using JSP

* It is very much convenient to modify the regular HTML.
* We can write the servlet code into the JSP.
* It is only intended for simple inclusions which can use form data and make connections.
* You can easily connect with JSP with the MySQL database.
* The performance and scalability of JSP are good because JSP allows embedding of dynamic elements in HTML pages.
* JSP is built on Java technology hence it is platform-independent and not depending on any operating systems.
* JSP includes the feature of multithreading of java.
* We can also make use of exception handling of java into JSP.
* It enables to separate presentation layer with the business logic layer in the web application.
* It is easy for developers to show as well as process the information.

## Disadvantages of JSP

Here are cons/drawbacks for using JSP:

* It is hard to trace JSP pages error because JSP pages are translated to servlet.
* As JSP output is HTML, it is not rich in features.
* It is very hard to debug or trace errors because JSP pages are first translated into servlets before the compilation process.
* Database connectivity is not easy.
* JSP pages require more disk space and time to hold JSP pages as they are compiled on the server.

AJAX

AJAX stands for **A**synchronous **Ja**vaScript and **X**ML. AJAX is a new technique for creating better, faster, and more interactive web applications with the help of XML, HTML, CSS, and Java Script.

* Ajax uses XHTML for content, CSS for presentation, along with Document Object Model and JavaScript for dynamic content display.
* Conventional web applications transmit information to and from the sever using synchronous requests. It means you fill out a form, hit submit, and get directed to a new page with new information from the server.
* With AJAX, when you hit submit, JavaScript will make a request to the server, interpret the results, and update the current screen. In the purest sense, the user would never know that anything was even transmitted to the server.
* XML is commonly used as the format for receiving server data, although any format, including plain text, can be used.
* AJAX is a web browser technology independent of web server software.
* A user can continue to use the application while the client program requests information from the server in the background.
* Intuitive and natural user interaction. Clicking is not required, mouse movement is a sufficient event trigger.
* Data-driven as opposed to page-driven.
* **AJAX allows you to send and receive data asynchronously without reloading the web page. So it is fast.**
* **AJAX allows you to send only important information to the server not the entire page. So only valuable data from the client side is routed to the server side. It makes your application interactive and faster.**

## **AJAX is Based on Open Standards**

AJAX is based on the following open standards −

* Browser-based presentation using HTML and Cascading Style Sheets (CSS).
* Data is stored in XML format and fetched from the server.
* Behind-the-scenes data fetches using XMLHttpRequest objects in the browser.
* JavaScript to make everything happen.

### Where it is used?

There are too many web applications running on the web that are using ajax technology like **gmail**, **facebook**,**twitter**,**google map**, **youtube** etc.

# **Ajax Java Example**

To create [ajax](https://www.javatpoint.com/ajax-tutorial) example, you need to use any server-side language e.g. [Servlet](https://www.javatpoint.com/servlet-tutorial), [JSP](https://www.javatpoint.com/jsp-tutorial), [PHP](https://www.javatpoint.com/php-tutorial), [ASP.Net](https://www.javatpoint.com/asp-net-tutorial) etc. Here we are using JSP for generating the server-side code.

In this example, we are simply printing the table of the given number.

#### Steps to create ajax example with jsp

You need to follow following steps:

1. load the org.json.jar file
2. create input page to receive any text or number
3. create server side page to process the request
4. provide entry in web.xml file

#### Load the org.json.jar file

download this example, we have included the org.json.jar file inside the WEB-INF/lib directory.

#### create input page to receive any text or number

In this page, we have created a form that gets input from the user. When user clicks on the showTable button, **sendInfo()** function is called. We have written all the ajax code inside this function.

We have called the **getInfo()** function whenever ready state changes. It writes the returned data in the web page dynamically by the help of **innerHTML** property.

**table1.html**

1. **<html>**
2. **<head>**
3. **<script>**
4. var request;
5. function sendInfo()
6. {
7. var v=document.vinform.t1.value;
8. var url="index.jsp?val="+v;
10. if(window.XMLHttpRequest){
11. request=new XMLHttpRequest();
12. }
13. else if(window.ActiveXObject){
14. request=new ActiveXObject("Microsoft.XMLHTTP");
15. }
17. try
18. {
19. request.onreadystatechange=getInfo;
20. request.open("GET",url,true);
21. request.send();
22. }
23. catch(e)
24. {
25. alert("Unable to connect to server");
26. }
27. }
29. function getInfo(){
30. if(request.readyState==4){
31. var val=request.responseText;
32. document.getElementById('amit').innerHTML=val;
33. }
34. }
36. **</script>**
37. **</head>**
38. **<body>**
39. **<marquee><h1>**This is an example of ajax**</h1></marquee>**
40. **<form** name="vinform"**>**
41. **<input** type="text" name="t1"**>**
42. **<input** type="button" value="ShowTable" onClick="sendInfo()"**>**
43. **</form>**
45. **<span** id="amit"**>** **</span>**
47. **</body>**
48. **</html>**

create server side page to process the request

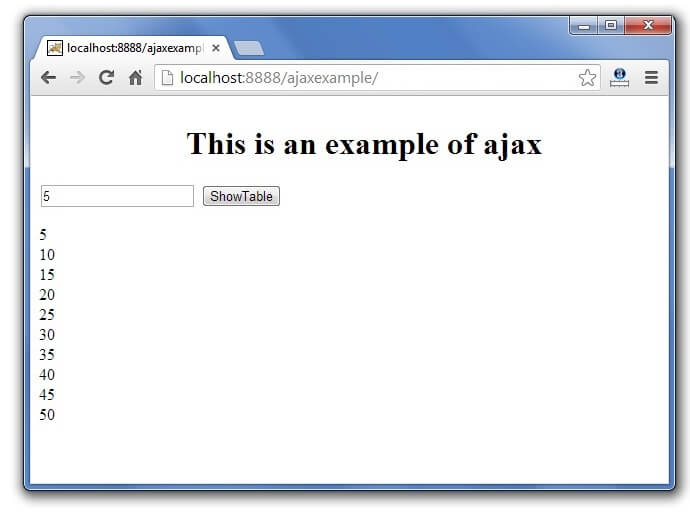
In this jsp page, we printing the table of given number.

**index.jsp**

1. **<**%
2. int n=Integer.parseInt(request.getParameter("val"));
4. for(int i=1;i**<**=10;i++)
5. out.print(i\*n+"**<br>**");
7. %**>**

web.xml

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. **<web-app** version="2.5" xmlns="http://java.sun.com/xml/ns/javaee"
3. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4. xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
5. http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"**>**
7. **<session-config>**
8. **<session-timeout>**
9. 30
10. **</session-timeout>**
11. **</session-config>**
12. **<welcome-file-list>**
13. **<welcome-file>**table1.html**</welcome-file>**
14. **</welcome-file-list>**
15. **</web-app>**



## **What is JSF?**

**JavaServer Faces** (JSF) is a MVC web framework that simplifies the construction of User Interfaces (UI) for server-based applications using reusable UI components in a page. JSF provides a facility to connect UI widgets with data sources and to server-side event handlers. The JSF specification defines a set of standard UI components and provides an Application Programming Interface (API) for developing components. JSF enables the reuse and extension of the existing standard UI components.

## **Benefits**

JSF reduces the effort in creating and maintaining applications, which will run on a Java application server and will render application UI on to a target client. JSF facilitates Web application development by −

* Providing reusable UI components
* Making easy data transfer between UI components
* Managing UI state across multiple server requests
* Enabling implementation of custom components
* Wiring client-side event to server-side application code

## **JSF UI Component Model**

JSF provides the developers with the capability to create Web application from collections of UI components that can render themselves in different ways for multiple client types (for example - HTML browser, wireless, or WAP device).

JSF provides −

* Core library
* A set of base UI components - standard HTML input elements
* Extension of the base UI components to create additional UI component libraries or to extend existing components
* Multiple rendering capabilities that enable JSF UI components to render themselves differently depending on the client types

# **JavaServer Faces Lifecycle**

JavaServer Faces application framework manages lifecycle phases automatically for simple applications and also allows you to manage that manually. The lifecycle of a JavaServer Faces application begins when the client makes an HTTP request for a page and ends when the server responds with the page.

The JSF lifecycle is divided into two main phases:

1. Execute Phase
2. Render Phase

## **1) Execute Phase**

In execute phase, when first request is made, application view is built or restored. For other subsequent requests other actions are performed like request parameter values are applied, conversions and validations are performed for component values, managed beans are updated with component values and application logic is invoked.

The execute phase is further divided into following subphases.

* Restore View Phase
* Apply Request Values Phase
* Process Validations Phase
* Update Model Values Phase
* Invoke Application Phase
* Render Response Phase

### Restore View Phase

When a client requests for a JavaServer Faces page, the JavaServer Faces implementation begins the restore view phase. In this phase, JSF builds the view of the requested page, wires event handlers and validators to components in the view and saves the view in the FacesContext instance.

If the request for the page is a postback, a view corresponding to this page already exists in the FacesContext instance. During this phase, the JavaServer Faces implementation restores the view by using the state information saved on the client or the server.

Apply Request Values Phase

In this phase, component tree is restored during a postback request. Component tree is a collection of form elements.Each component in the tree extracts its new value from the request parameters by using its decode (processDecodes()) method. After that value is stored locally on each component.

* If any decode methods or event listeners have called the renderResponse method on the current FacesContext instance, the JavaServer Faces implementation skips to the Render Response phase.
* If any events have been queued during this phase, the JavaServer Faces implementation broadcasts the events to interested listeners.
* If the application needs to redirect to a different web application resource or generate a response that does not contain any JavaServer Faces components, it can call the FacesContext.responseComplete() method.
* If the current request is identified as a partial request, the partial context is retrieved from the FacesContext, and the partial processing method is applied.

Process Validations Phase

In this phase, the JavaServer Faces processes all validators registered on the components by using its validate () method. It examines the component attributes that specify the rules for the validation and compares these rules to the local value stored for the component. The JavaServer Faces also completes conversions for input components that do not have the immediate attribute set to true.

* If any validate methods or event listeners have called the renderResponse method on the current FacesContext, the JavaServer Faces implementation skips to the Render Response phase.
* If the application needs to redirect to a different web application resource or generate a response that does not contain any JavaServer Faces components, it can call the FacesContext.responseComplete method.
* If events have been queued during this phase, the JavaServer Faces implementation broadcasts them to interested listeners.
* If the current request is identified as a partial request, the partial context is retrieved from the FacesContext, and the partial processing method is applied.

Update Model Values Phase

After ensuring that the data is valid, it traverses the component tree and sets the corresponding server-side object properties to the components' local values. The JavaServer Faces implementation updates only the bean properties pointed at by an input component's value attribute. If the local data cannot be converted to the types specified by the bean properties, the lifecycle advances directly to the Render Response phase so that the page is re-rendered with errors displayed.

* If any updateModels methods or any listeners have called the renderResponse() method on the current FacesContext instance, the JavaServer Faces implementation skips to the Render Response phase.
* If the application needs to redirect to a different web application resource or generate a response that does not contain any JavaServer Faces components, it can call the FacesContext.responseComplete() method.
* If any events have been queued during this phase, the JavaServer Faces implementation broadcasts them to interested listeners.
* If the current request is identified as a partial request, the partial context is retrieved from the FacesContext, and the partial processing method is applied.

Invoke Application Phase

In this phase, JSF handles application-level events, such as submitting a form or linking to another page.

Now, if the application needs to redirect to a different web application resource or generate a response that does not contain any JSF components, it can call the FacesContext.responseComplete() method.

After that, the JavaServer Faces implementation transfers control to the Render Response phase.

Render Response Phase

This is last phase of JSF life cycle. In this phase, JSF builds the view and delegates authority to the appropriate resource for rendering the pages.

* If this is an initial request, the components that are represented on the page will be added to the component tree.
* If this is not an initial request, the components are already added to the tree and need not to be added again.
* If the request is a postback and errors were encountered during the Apply Request Values phase, Process Validations phase, or Update Model Values phase, the original page is rendered again during this phase.
* If the pages contain h:message or h:messages tags, any queued error messages are displayed on the page.
* After rendering the content of the view, the state of the response is saved so that subsequent requests can access it. The saved state is available to the Restore View phase.

## **2) Render**

In this phase, the requested view is rendered as a response to the client browser. View rendering is a process in which output is generated as HTML or XHTML. So, user can see it at the browser.

The following steps are taken during the render process.

* Application is compiled, when a client makes an initial request for the index.xhtml web page.
* Application executes after compilation and a new component tree is constructed for the application and placed in a FacesContext.
* The component tree is populated with the component and the managed bean property associated with it, represented by the EL expression.
* Based on the component tree. A new view is built.
* The view is rendered to the requesting client as a response.
* The component tree is destroyed automatically.
* On subsequent requests, the component tree is rebuilt, and the saved state is applied.