**Introduction:**

JavaServer Faces technology is a server-side user interface component framework for Java

technology-based web applications.

The main components of JavaServer Faces technology are as follows:

* An API for representing UI components and managing their state; handling events,

server-side validation, and data conversion; defining page navigation; supporting

internationalization and accessibility; and providing extensibility for all these features

■ Two JavaServer Pages (JSP) custom tag libraries for expressing UI components within a JSP

page and for wiring components to server-side objects.

* The well-defined programming model and tag libraries significantly ease the burden of building and maintaining web applications with server-side UIs.
* With minimal effort, you can Drop components onto a page by adding component tags

■ Wire component-generated events to server-side application code

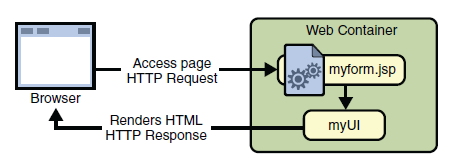
■ Bind UI components on a page to server-side data

■ Construct a UI with reusable and extensible components

■ Save and restore UI state beyond the life of server requests

**JavaServer FacesTechnologyUser Interface**

the user interface you create with JavaServer Faces technology runs on the server and renders back to the client.



The JSP page, myform.jsp, is a *JavaServer Faces page*, which is a JSP page that includes

JavaServer Faces tags. It expresses the user interface components by using custom tags defined

by JavaServer Faces technology. The UI for the web application (represented by myUI in the

figure) manages the objects referenced by the JSP page. These objects include

■ The UI component objects that map to the tags on the JSP page

■ Any event listeners, validators, and converters that are registered on the components

■ The JavaBeans components that encapsulate the data and application-specific functionality

of the components

**JavaServer FacesTechnology Benefits**

One of the greatest advantages of JavaServer Faces technology is that it offers a clean separation

between behavior and presentation.

The separation of logic from presentation also allows each member of a web application

development team to focus on his or her piece of the development process, and it provides a

simple programming model to link the pieces.

Although JavaServer Faces technology includes a JSP custom tag library for

representing components on a JSP page, the JavaServer Faces technology APIs are layered

directly on top of the Servlet API, This layering of APIs enables several

important application use cases, such as using another presentation technology instead of JSP

pages, creating your own custom components directly from the component classes, and

generating output for various client devices.

Most importantly, JavaServer Faces technology provides a rich architecture for managing

component state, processing component data, validating user input, and handling events.

**What Is a JavaServer Faces Application?**

For the most part, a JavaServer Faces application is like any other Java web application. A typical

JavaServer Faces application includes the following pieces:

■ A set of JSP pages (although you are not limited to using JSP pages as your presentation

technology)

■ A set of *backing beans*, which are JavaBeans components that define properties and

functions for UI components on a page

■ An application configuration resource file, which defines page navigation rules and

configures beans and other custom objects, such as custom components

■ A deployment descriptor (a web.xml file)

■ Possibly a set of custom objects created by the application developer. These objects might

include custom components, validators, converters, or listeners.

■ A set of custom tags for representing custom objects on the page

A JavaServer Faces application that includes JSP pages also uses the standard tag libraries

defined by JavaServer Faces technology for representing UI components and other objects on

the page.

**Steps in the Development Process**

Developing a simple JavaServer Faces application usually requires these tasks:

■ Mapping the FacesServlet instance.

■ Creating the pages using the UI component and core tags.

■ Defining page navigation in the application configuration resource file.

■ Developing the backing beans.

■ Adding managed bean declarations to the application configuration resource file.

**Mapping the** FacesServlet **Instance**

All JavaServer Faces applications must include a mapping to the FacesServlet instance in their

deployment descriptors. The FacesServlet instance accepts incoming requests, passes them to

the life cycle for processing, and initializes resources.

Example:

<servlet>

<display-name>FacesServlet</display-name>

<servlet-name>FacesServlet</servlet-name>

<servlet-class>javax.faces.webapp.FacesServlet</servlet-class>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>FacesServlet</servlet-name>

<url-pattern>/guess/\*</url-pattern>

</servlet-mapping>