

#### Get Involved

[java-net Project](#)  
[Request a Project](#)  
[Project Help Wanted Ads](#)  
[Publicize your Project](#)  
[Submit Content](#)

#### Get Informed

[About java.net](#)  
[Articles](#)  
[Weblogs](#)  
[News](#)  
[Events](#)  
[Also in Java Today](#)  
[java.net Online Books](#)  
[java.net Archives](#)

#### Get Connected

[java.net Forums](#)  
[Wiki and Javapedia](#)  
[People, Partners, and Jobs](#)  
[Java User Groups](#)  
[RSS Feeds](#)

#### Search

Web and Projects:

»

Online Books:

»

[Advanced Search](#)

## FREQUENTLY ASKED QUESTIONS about JavaServer Faces Technology

### General Questions

- [What is JavaServer Faces technology?](#)
- [What are the benefits of JavaServer Faces technology?](#)
- [Who are the intended users of JavaServer Faces technology?](#)
- [Who is involved in developing the JavaServer Faces specification?](#)
- [What are the differences between JavaServer Faces technology and Struts?](#)
- [How does JavaServer Faces technology relate to JavaServer Pages \(JSP\) technology?](#)
- [Where can I find out more about JavaServer Faces technology?](#)

### JSF 1.1 Troubleshooting Questions

- [Do JavaServer Faces tags interoperate with JSTL core tags, forEach, if, choose and when?](#)
- [When I use <c:import> or <jsp:include> to include content from a JSP page into a JavaServer Faces page, it produces incorrect HTML output. Why does this happen?](#)

### JSF 1.2 Troubleshooting Questions

- [Do JavaServer Faces tags interoperate with JSTL core tags, forEach, if, choose and when?](#)
- [When I use <c:import> or <jsp:include> to include content from a JSP page into a JavaServer Faces page, it produces incorrect HTML output. Why does this happen?](#)

### For More Information

- [Is there a public chat room for JSF users and developers?](#)
- [Is there a Sun Private chat room for JSF users and developers inside of Sun?](#)
- [What are some real world sites using JSF?](#)

### General Questions

#### Q: What is JavaServer Faces technology?

A: JavaServer Faces technology is a framework for building user interfaces for web applications. JavaServer Faces technology includes:

- A set of APIs for: representing UI components and managing their state, handling events and input validation, defining page navigation, and supporting internationalization and accessibility.
- A JavaServer Pages (JSP) custom tag library for expressing a JavaServer Faces interface within a JSP page.

With the simple, well-defined programming model that JavaServer Faces technology provides, developers of varying skill levels can quickly and easily build Web applications by: assembling reusable UI components in a page, connecting these components to an application data source, and wiring client-generated events to server-side event handlers. With the power of JavaServer Faces technology, these web applications handle all of the complexity of managing the user interface on the server, allowing the application developer to focus on their application code.

#### Q: What are the benefits of JavaServer Faces technology?

A: The primary benefits of JavaServer Faces technology include:

- **Ease-of-Use:** Several aspects of the JavaServer Faces architecture contribute to its ease-of-use. For one, JavaServer Faces technology offers a clean separation between logic and presentation, enabling a wide range of users -- from web-page designers to component developers-- to take advantage of JavaServer technology, resulting in a division of labor and a shorter development cycle. Also, a user-interface created with JavaServer Faces technology handles all the complexities of user-interface management, including input validation, component-state management, page navigation, and event handling.
- **Standardization:** JavaServer Faces technology is being developed through the Java Community Process. Several prominent, respected tools vendors are contributing members of the expert group and are committed to supporting JavaServer Faces technology in their tools.
- **Device Independence:** JavaServer Faces technology is designed to be flexible. By defining only component functionality in extensible UI component classes, the JavaServer Faces architecture allows component developers to extend the component classes to generate their own component tag libraries targetted for specific clients.

#### Q: Who are the intended users of JavaServer Faces technology?

A: Because of the extensibility and ease-of-use that JavaServer Faces technology provides, a wide range of developers and web-page designers can take advantage of the power of JavaServer Faces technology. These users include:

- **Page Authors**, who build the UI using JavaServer Faces component tags from within web pages, such as JSP pages. This set of users will most likely be the primary users of the JavaServer Faces custom tag library.
- **Application Developers**, who write the application code, including the data-access, event-handling, and business logic.
- **Component Writers**, who construct reusable UI components, and will take advantage of the extensibility of the UI component classes to build custom components that can be targeted for a specific client.
- **Tools Vendors**, who build tools leveraging JavaServer Faces technology to make building a user interface with JavaServer Faces technology even easier.

**Q: Who is involved in developing the JavaServer Faces specification?**

A: A number of important industry players are collaborating with Sun to define the first draft of the JavaServer Faces specification. Please see [JSR-252](#) for a complete list of expert group members.

**Q: What are the differences between JavaServer Faces technology and Struts?**

A: Struts is an open-source Java web application framework whose architecture is based on the [Model-View-Controller](#) design pattern in which requests are routed through a controller that provides overall application management and dispatches the requests to application components. JavaServer Faces technology is a user-interface framework for Java web applications. It is focussed on the view tier of an MVC-based architecture. The Struts and JavaServer Faces technology frameworks do have some overlapping functionality; however each framework has its advantages, and developers can use certain features of both frameworks in a single application.

The primary advantages of Struts as compared to JavaServer Faces technology are as follows:

- Because Struts is a web application framework, it has a more sophisticated controller architecture than does JavaServer Faces technology. It is more sophisticated partly because the application developer can access the controller by creating an Action object that can integrate with the controller, whereas JavaServer Faces technology does not allow access to the controller. In addition, the Struts controller can do things like access control on each Action based on user roles. This functionality is not provided by JavaServer Faces technology.
- Struts includes a powerful layout management framework, called Tiles, which allows you to create templates that you can reuse across multiple pages, thus enabling you to establish an overall look-and-feel for an application.
- The Struts validation framework includes a larger set of standard validators, which automatically generate both server-side and client-side validation code based on a set of rules in a configuration file. You can also create custom validators and easily include them in your application by adding definitions of them in your configuration file.

The greatest advantage that JavaServer Faces technology has over Struts is its flexible, extensible UI component model, which includes:

- A standard component API for specifying the state and behavior of a wide range of components, including simple components, such as input fields, and more complex components, such as scrollable data tables. Developers can also create their own components based on these APIs, and many third parties have already done so and have made their component libraries publicly available.
- A separate rendering model that defines how to render the components in various ways. For example, a component used for selecting an item from a list can be rendered as a menu or a set of radio buttons.
- An event and listener model that defines how to handle events generated by activating a component, such as what to do when a user clicks a button.
- Conversion and validation models for converting and validating component data.

Because the JavaServer Faces technology architecture separates the definition of a component from its rendering, you can render your components in different ways or even to different clients, such as a WML client. Moreover, the extensible component APIs of JavaServer Faces technology allow you to extend the standard set of components and create entirely new components. None of this is possible with Struts. In fact, Struts has no notion of server-side components, which also means that it has no event model for responding to component events and no facility for saving and restoring component state. While Struts does have a useful tag library for rendering components on the page, these components have no object representation on the server and they can only be rendered to an HTML client.

Another distinct advantage of JavaServer Faces technology is that it is standard, which means that it has been developed through the Java Community Process (JCP) and has been designed to allow easy integration into tools. As a result, JavaServer Faces technology already has wide industry support and is being leveraged by several web application development IDEs (such as [Sun Java Studio Creator](#)).

Because both JavaServer Faces technology and Struts contribute such valuable features, developers might want to be able to use both of them in a single application. Developers might want to integrate the flexible component model of JavaServer Faces technology into their existing Struts applications while continuing to use the Struts controller architecture. Similarly, developers who have JavaServer Faces technology applications might want to integrate the more powerful client-side validation mechanism and Tiles layout framework found in the Struts architecture into their applications. These goals can be accomplished by using the struts-faces integration library, which you can download from [here](#).

For more information on the relationship of Struts and JavaServer Faces technology, see Craig McClanahan's [blog entry](#) on this topic.

A successor to Struts is also just getting started. It's called Shale, and you can learn more here:

<http://www.apache.org/archives/000552.html>. Since Shale is also an open-source project, people interested in participating in the construction of this framework are welcome to join. You can subscribe to the wiki page here: <http://wiki.apache.org/struts/StrutsShale>.

**Q: How does JavaServer Faces technology relate to JavaServer Pages (JSP) technology?**

A: JavaServer Faces technology, version 1.0 relies on JSP 1.2. Since JSP 2.0 is a superset of JSP 1.2, it is possible to use JavaServer Faces technology, version 1.0 with JSP 2.0. Future versions of the JavaServer Faces specification will be able to take

better advantage of JSP 2.0.

**Q: Where can I find out more about JavaServer Faces technology?**

A: See chapters 9 through 13 of the [Java EE 5 Tutorial](#) to learn more about JavaServer Faces technology. If you have additional questions not answered by this FAQ, consult the [JavaServer Faces Forum](#).

## JSF 1.1 Troubleshooting Questions

Q. Do JavaServer Faces tags interoperate with JSTL core tags, `forEach`, `if`, `choose` and `when`?

A. The `forEach` tag does not work with JavaServer Faces technology, version 1.0 and 1.1 tags due to an incompatibility between the strategies used by JSTL and JavaServer Faces technology. Instead, you could use a renderer, such as the `Table` renderer used by the `dataTable` tag, that performs its own iteration. The `if`, `choose`

and `when` tags work, but the JavaServer Faces tags nested within these tags must have explicit identifiers.

This shortcoming has been fixed in JSF 1.2.

Q. When I use `<c:import>` or `<jsp:include>` to include content from a JSP page into a JavaServer Faces page, it produces incorrect HTML output. Why does this happen?

A. When using `<jsp:include>` or `<c:import>` to compose a single view from multiple JSP pages, all JavaServer Faces component tags in the included pages must be nested inside the `<f:subview>` tag the JavaServer Faces core tag library (which is itself nested inside the `<f:view>` tag). The `<f:subview>`

tag itself can be present in the including page with the `<jsp:include>` or `<c:import>` tag nested inside it, or it can be in the included page.

Any template text or non-JavaServer Faces tags present in a page that is included with the `<jsp:include>` or `<c:import>` tag, or any other mechanism that uses `RequestDispatcher.include`, must be enclosed in an `<f:verbatim>` tag. This restriction has been lifted for JSF 1.2.

## JSF 1.2 Troubleshooting Questions

Q. Do JavaServer Faces tags interoperate with JSTL core tags, `forEach`, `if`, `choose` and `when`?

A. Yes. A new feature of JSP 2.1, called [JSP Id Consumer](#) allows these tags to work as expected.

Q. When I use `<c:import>` or `<jsp:include>` to include content from a JSP page into a JavaServer Faces page, it produces incorrect HTML output. Why does this happen?

A. This has been fixed in JSF 1.2

## For More Information

- Is there a public chat room for JSF developers and users?

Absolutely! We use `##jsf` on [irc.freenode.net](#). Converse with members of the team working on JavaServer Faces technology by joining our public chat room on freenode.net. After you access freenode with your IRC client for the first time, you register yourself with the following command:

```
/msg NickServ REGISTER <your password>
```

The password should be one that you don't mind others accessing. In other words, don't use a password that you already use for something else important.

Every time you log in thereafter, you need to identify yourself to the NickServ:

```
/msg NickServ IDENTIFY <your password>
```

Finally, after you've registered your password or identified yourself, you join the channel:

```
/join ##jsf
```

(Note the two `##`). Happy chatting!

- Is there a Sun Private public chat room for JSF developers and users inside of Sun?


Absolutely! We use `#jsfaces` on [Sun's internal IRC network](#). Change the domain name to `irc.>yourdomain<.sun.com` where `yourdomain` is `east`, `sfbay`, `central`, etc. Converse with members of the team working on JavaServer Faces technology by joining our public chat room on Sun's Intranet. After you access freenode with your IRC client for the first time, you join the channel with the following command:

/join #jsfaces

Happy chatting!

Here is a wiki page that collects [links to real world sites using JSF](#).

Topic **JavaServerFacesSpecFaq** . { [Edit](#) | [Ref-By](#) | [Printable](#) | [Diffs](#) [r6](#) < [r5](#) < [r4](#) < [r3](#) < [r2](#) | [More](#) }

 [java.net RSS](#)



[Feedback](#) | [FAQ](#) | [Terms of Use](#)  
[Privacy](#) | [Trademarks](#) | [Site Map](#)

Your use of this web site or any of its content or software indicates your agreement to be bound by these [Terms of Participation](#).

Copyright © 1995-2006 Sun Microsystems, Inc.

**O'REILLY** **COLLABNET**

Powered by Sun Microsystems, Inc.,  
O'Reilly and [CollabNet](#)

Revision [r6](#) - 11 Jul 2006 - 03:37:27 - Main.gojira  
Parents: [WebHome](#) > [Javaserverfaces](#)

TWiki (TM) Copyright © 1999-2003 by Peter Thoeny ALL RIGHTS  
RESERVED