The key differences between the two are the base paradigms that underly each platform.

Specifically, JSF is a "component" framework whereas Struts is an "action" framework.

What does that mean?

In a component framework, artifacts that are rendered on the page are initially developed as individual components, much like in modern GUI "fat client" libraries. You have components, they have events, and your code is written to work with those events against the components.

Most of the time, in mainstream development, your code is pretty much ignorant of the HTTP request cycle and processing.

Struts (both 1 and 2) are action frameworks. In essence they give you the ability to map URLs to activities and code on the back end. Here, the layout and workflow tends to be more page oriented. As a developer you tend to interact with the HTTP request cycle directly, though Struts 2 helps isolate at least the binding of the request data to the action implementation classes.

Action frameworks tend to be much "thinner" in how they stand between your code and the raw HTTP request compared to component frameworks.

For those people just cutting their teeth on web development, the component frameworks (especially with good tooling) can be much more approachable, as the tools tend to hide the heavy weight nature of the component frameworks. But, on the downside, their size and their "square peg in to round hole" nature of turning HTTP requests in to the rough equivalent of mouse clicks mean there's more complexity to try to understand if and when the framework starts misbehaving or getting in the way.

But, to be fair, frameworks like ASP.NET and JSF are popular because novices can get quick success with them with the modern tools.

Old School HTTP wranglers may simply be more comfortable with action frameworks, simply because they've been through the crucible of understanding how HTTP requests are structured. Action frameworks tend to work better with "pretty" urls (though component frameworks are getting better at his). Action framework coders can have more control of the structure and presentation of URLs, since their systems are more intimately tied to them compared to a component framework.

As a basic guideline, I find the action frameworks are better for "web sites", site like this one, sites that focus on delivering content to the user. Where it's mostly a "read only" experience for the end user who is likely to want to bookmark things, come back to arbitrarily deep pages, etc.

But the component frameworks are better for "web apps". CRUD screens, back office applications, lots of forms and controls, etc. Here, the workflow is more controlled. You tend to not get to a "detail" screen with going through the "list" screen or "header" screen first, for example.

It's nice to be able to just drag and drop a grid component on to a form, add a couple of buttons and point it as a DB table to get "instant" results. But these apps don't bookmark well, HATE the "refresh button", may behave poorly with the back button, etc. Overall, they can not be very good citizens when working with the web browser.

So, if it were me, I'd write a blog in a action framework like Struts 2 (or even better, Stripes), but I'd write an accounting package in JSF.

[Struts](http://www.java-samples.com/struts/) is an open-source Java web application framework whose architecture is based on the [Model-View-Controller](http://www.java-samples.com/showtutorial.php?tutorialid=350) 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](http://www.java-samples.com/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](http://developers.sun.com/prodtech/javatools/jscreator/index.jsp)).  
  
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 stuts-faces integration library, which you can download from [here](http://svn.apache.org/builds/jakarta-struts/nightly/struts-faces/).

Both these Frameworks are used for forcing the

implementation of MVC-2 pattern.

Struts Framework is implemented by apache, where as JSF

is implemented by Sun Microsystems.

|  |  |  |
| --- | --- | --- |
| **Is This Answer Correct ?** | **71 Yes** | 27 No |

1

**Bhoopesh Alladi**   
    Re: **Answer**  
# [2](http://www.allinterview.com/viewpost/73784.html)

Struts is JSP and Tag centric and with a fixed dependecy on

being deployed in a conventional Servlet HTTP request /

response environment.

JSF is somewhat more generic. It is component centric and

has no specific dependency on HTML markup / tags apart from

the fact that the renders for most components are tailored

for conventional browser output via HTML.

|  |  |  |
| --- | --- | --- |
| **Is This Answer Correct ?** | **50 Yes** | 10 No |

2

**Meera Tolia**

|  |
| --- |
|  |

      Re: **Answer**  
# [3](http://www.allinterview.com/viewpost/75939.html)

both will implement mvc2 design pattern,and struts given by

apache ,jsf given by sun,instead of actionservlet we can

find facelet asa acontroller in jsf

|  |  |  |
| --- | --- | --- |
| **Is This Answer Correct ?** | **15 Yes** | 19 No |

3

**Karnakar**   
    Re: **Answer**  
# [4](http://www.allinterview.com/viewpost/89833.html)

Both will implement mvc2 design pattern, struts given by

apache, jsf given by sun.

Struts didn't provide any use implementations but JSF we

can implement our own implementations like UI-components,

JSF provides the Converters also

|  |  |  |
| --- | --- | --- |
| **Is This Answer Correct ?** | **13 Yes** | 10 No |

0

**Kiran**   
    Re: **Answer**  
# [5](http://www.allinterview.com/viewpost/104841.html)

The main difference is the UIComponent of JSF. Developers

can define their own components and also the way it should

be rendered. A component having a particular behavious can

be rendered in different ways.

Eg: UISelectOne. This components behaviour is to select one

value, but can be rendered as a List, Menu or Radio.

More we can develop RenderKit's appropriate to the client,

ie html, wml, etc. So a single JSF application can be used,

and using different renderers can be rendered to different

clients...

|  |  |  |
| --- | --- | --- |
| **Is This Answer Correct ?** | **20 Yes** | 4 No |

3

**Krishna**   
    Re: **Answer**  
# [6](http://www.allinterview.com/viewpost/125679.html)

JSF - Component centric; Struts - Bean centric.

JSF - Deffered expresson; Struts - Immediate expression.

JSF - Event driven; Struts - Model driven.

JSF - Value binding; Struts - Bean binding

JSF - Spec by JCP and several implementations including

SUN's; Struts - Apache.

|  |  |  |
| --- | --- | --- |
| **Is This Answer Correct ?** | **44 Yes** | 4 No |

3

**Anup**   
    Re: **Answer**  
# [7](http://www.allinterview.com/viewpost/200846.html)

Jsf is component based frame work which concentrates on

UIComponents look and feel(view layer) where as struts

concentrates more on the controller part not on the

userinterface but both follow model-2 architecture

|  |  |  |
| --- | --- | --- |
| **Is This Answer Correct ?** | **7 Yes** | 3 No |

0

**Ravi**   
    Re: **Answer**  
# [8](http://www.allinterview.com/viewpost/203913.html)

Configuration wise both are same and supports the

internalization. JSF is more developer friendly why because

to validate mandatory filed in JSF we can specify attribute

like require is true and we can use mandatory filed id in

message tag. But in struts we have to include

validation-rules.xml in struts-config.xml and form has to be

specified in validation-rules.xml.

Even for condition statement in JSF is better than struts.

No need to overwrite any method like execute in JSF.

navigation is very easy in JSF.

|  |  |  |
| --- | --- | --- |
| **Is This Answer Correct ?** | **9 Yes** | 2 No |

0

**Kollu Sreenivasa Rao**   
    Re: **Answer**  
# [9](http://www.allinterview.com/viewpost/216187.html)

Struts is a "Component" Framework, whereas Struts is

an "action" Framework.

JSF using Navigationcase and NavigationRules instead of

Actionmapping and action whic