

B E Y O N D

Think Beyond. Think Mphasis.

WHITE PAPER



## JSF – A Practical Evaluation

Mohan Borwankar

January, 2011

## Table of Contents

1	Summary .....	2
2	Overview of JSF .....	2
3	JSF/ Spring MVC / Struts – A Quick Comparison .....	3
4	JSF Prototype .....	6
5	Inputs from The Battlefield .....	6
6	Conclusion .....	7
	References .....	8
	Acknowledgements .....	8

## 1 Summary

Developing web based applications constitutes a major chunk of Java based projects. Organizations continuously explore frameworks and tools which would help improve the productivity of developers, achieving faster delivery. A large number of frameworks is available and a quick search on Google yields about half a million links for “Java based web application frameworks”. Lack of information affects the choice of framework; however a huge amount of information on the web is equally disastrous. Multiple sites echo conflicting opinions and usually the opinion depends on the principal sponsor of the site. Wikipedia has listed the 30 most commonly used Java Based Application frameworks<sup>1</sup>. Struts and Spring MVC were the most commonly used and JSF is the latest buzzword around.

A large number of projects are planning to use JSF either willingly or forced by the end customer. At the same time, a large number of resources are trained and skilled in Struts and Spring MVC. This is a conflict architects will have to face when making a decision on JSF.

The purpose of this whitepaper is to help the architects and technical leaders, who are the stakeholders in any technical decision, make an informed choice on JSF. Project Managers will find the learning of the project teams and pilot useful and help them plan JSF projects in a realistic way. This paper is based on experiences of MphasiS project teams, which implemented large scale JSF applications. The findings in this paper are also based on a pilot implementation of a JSF based application. The pilot was implemented by developers who were relatively new to JSF but experienced in Struts and Spring MVC. The pilot gave a very clear picture about the learning curve of JSF.

## 2 Overview of JSF

Java Server Faces (JSF) is yet another Java based Web Application Framework. Like other frameworks, this framework aims to simplify the development of user interfaces. Frameworks like Struts and Spring MVC are based on the request driven MVC model. JSF is a component based framework. The state of the UI is maintained on the server side and the state is refreshed every time any action is performed on the UI. JSF uses JSP to display the pages but can also use other technologies like XUL.

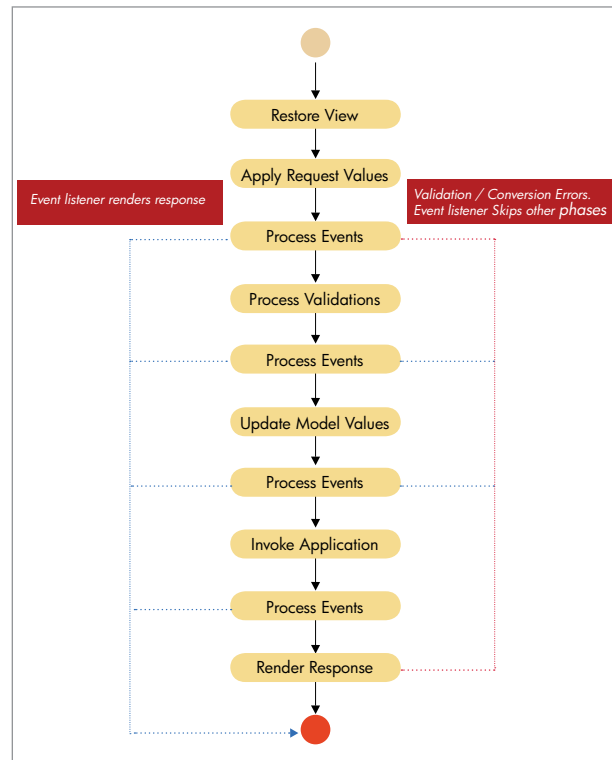
The JSF specification was developed under the Java Community Process as JSR 127, which defined JSF 1.0 and 1.1, and JSR 252 which defined JSF 1.2. Upcoming JSF 2.0 is being developed as JSR 314<sup>2</sup>.

### Request processing in JSF

The JSF framework is built on top of the Servlet API. All

page requests are first sent to the JSF Servlet. However, JSF extends the basic Servlet request processing and simplifies the programming with components, events and listeners.

The figure below depicts the request process flow within the JSF framework or how JSF processes the request and helps us understand how the framework masks the underlying request processing nature of Servlet API.



There are six primary phases or steps in the request processing and after most of the phases events are broadcast. Listeners receive the events. The listener either performs application logic or manipulates the components. The listeners can pass the control back to the main process flow or jump to the last step skipping further processing. The listener can also directly render the output skipping the entire processing.

The first phase is “Restore View”. In a view all the components on a page are represented in a tree. The view can be stored on the client side (usually as a hidden field on the browser) or on the server side (in the session). When a JSF application receives a request from an existing page, it has to figure out what page sent the request so that it can identify which events the user generated, and then associate these events with the components on the sending page. This is the main job of the Restore View phase—to find the most recent view and apply the user’s input to it. Restoring the view also ensures that the component’s values, as well as any event listeners, validators, or converters associated with components in the tree, are restored.

The second phase is “Apply Request Values”. During this phase, the framework sets submitted values based on the

1 [http://en.wikipedia.org/wiki/List\\_of\\_web\\_application\\_frameworks#java](http://en.wikipedia.org/wiki/List_of_web_application_frameworks#java)

2 <http://java.sun.com/javase/javaserverfaces/download.html>

parameters sent in the request. After this phase, any existing events are broadcast to the interested listeners. As discussed earlier, the listener can either short circuit the entire process or pass the control back to the main process flow.

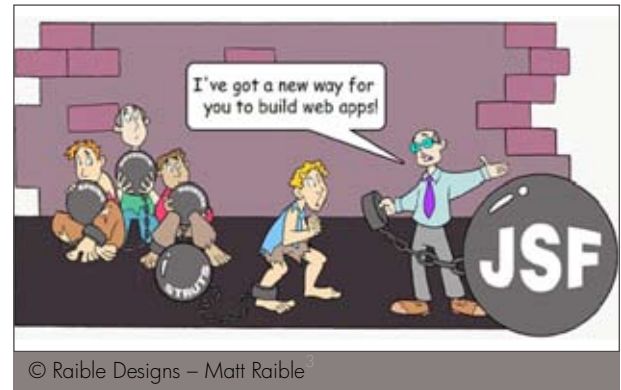
The third phase is “Process Validation”. In this phase, the entire component tree is traversed and each component is asked to verify if the submitted value is “Valid” or acceptable. Before the validation, the converter registered for each component converts the data. The converter can be custom or default. Validation is handled by the component or it can be delegated to one or more validators. Once all the conversions and validations are successful for all components, the control is passed to the next phase. In case of any error, the control may be transferred to the Render response phase to display error messages. Various events like date model event and value change event are broadcast during this phase.

The fourth phase is “Update Model Values”. At this point, the local values of components are converted and validated. The backing beans or the data objects are updated with these values. Once this is done, events are broadcast to interested listeners. By this point in the lifecycle, the user inputs are validated and component values are updated. All associated backing beans are also updated. All of this is achieved without invoking any application code. This is the main advantage of JSF – a lot of UI processing tasks are taken care of automatically.

The fifth phase is “Invoke Application”. At this juncture, all the necessary backing beans and model objects are already up to date and its time to invoke the core application functionality. In this phase, the JSF broadcasts events for all the registered listeners. Once all the listeners have executed, JSF is ready to display the response to the user.

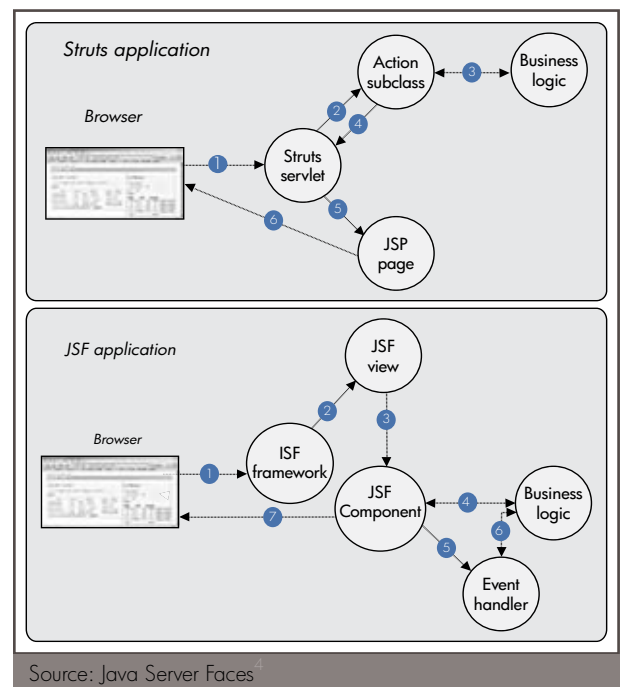
The final phase is “Render Response”. At this point all processing by the framework and the application has taken place. In this phase, the response is sent back to the user. The secondary activity in this phase is to save the state of the page so that it can be restored during the next processing request. JSF isn’t tied to a particular display technology and in this phase the output is generated accordingly. The web container will transmit this information to the browser for rendering.

### 3 JSF/ Spring MVC / Struts – A quick comparison



The picture above is sure to dampen the spirits on anyone who wishes to explore and experiment with JSF. However, this section does an objective comparison of JSF, Spring MVC and Struts based on certain key parameters. The next section looks at the architectural difference between JSF and Struts or Spring MVC.

The figure below depicts the basic difference between JSF and Struts during the invocation of framework and processing the request and rendering the response. Struts and Spring MVC have a similar flow.

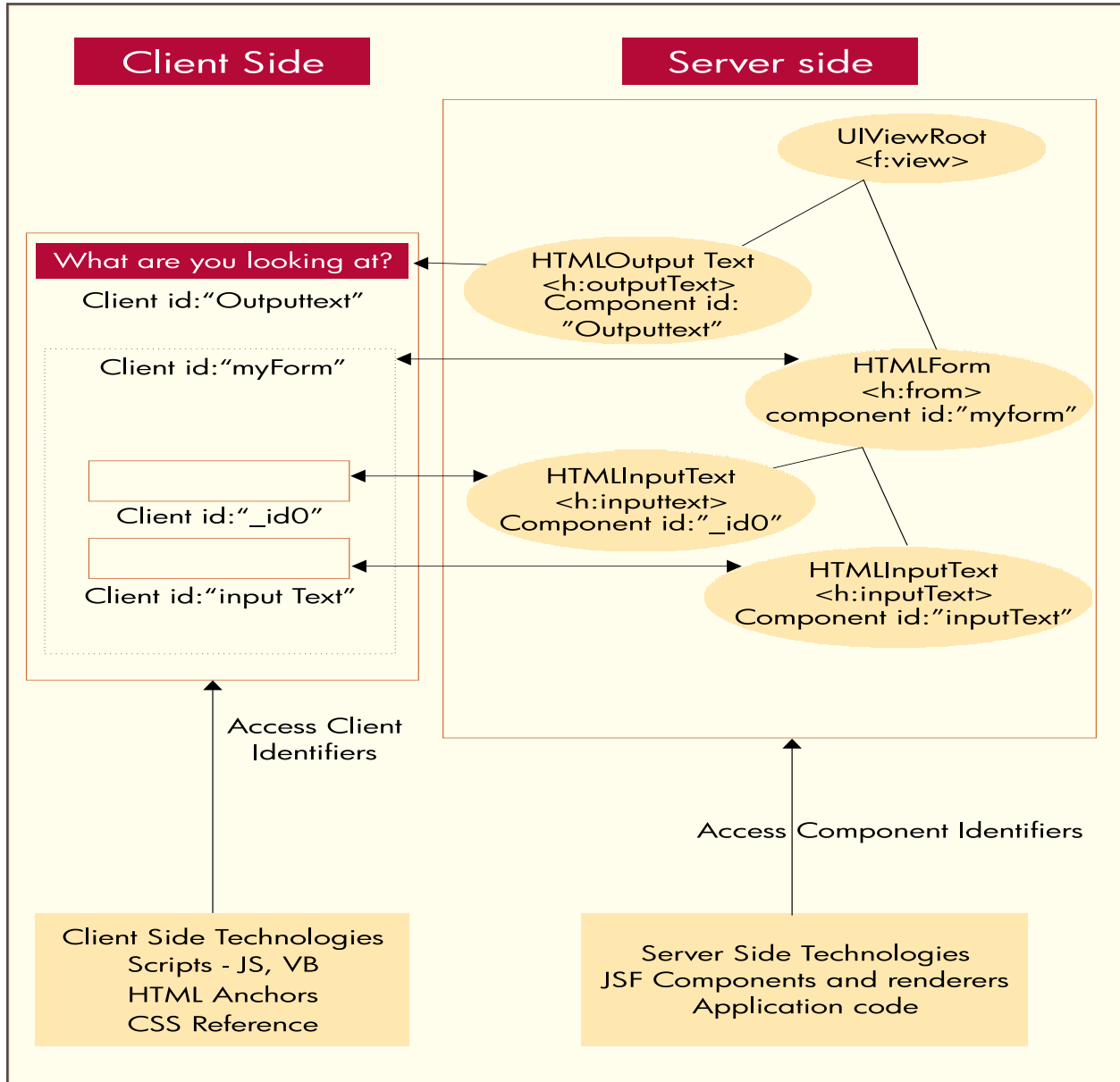


The Struts and Spring MVC frameworks are action driven. The action subclass associated with an action is invoked by the Struts Servlet. The action subclass invokes the business logic and returns the necessary data to the Servlet. The Servlet then renders the response.

3 <http://static.raibledesigns.com/repository/presentations/ComparingjavaWebFrameworks-ApacheConUS2007.pdf>  
 4 Java Server Faces – Hans Bergsten – O'Reilly.

The JSF framework maintains the state of the application on the server side. The entire UI view is recreated on the server side in a tree structure. The framework traverses the component tree and invokes all appropriate event handlers.

The components render the response. The figure below gives an idea about mapping the client side page onto the server side view. At first glance, the JSF server side view does look a little daunting.



Let's now have a closer look at the differences between JSF versus Struts or Spring MVC.

JSF is a relatively new framework. It addresses a few shortcomings of earlier frameworks; it is built on a totally different architecture. JSF adheres to the J2EE standards and is relatively fast and easy to develop with initially. The architecture is component based and there are a lot of component libraries available to choose from. However, there is no single source available for implementation and, being relatively new, the initial version had a few issues.

JSP is a proven technology for mixing static and dynamic content. JSP uses a tag library and extends the tag library with a custom library to enhance the display capabilities. As compared to JSP, JSF has a relatively complex lifecycle. Components are created, initialized, asked to process input and then rendered. This order of processing is a must in JSF. When JSP and JSF are used together, component creation and rendering happens in parallel and this causes a lot of problems<sup>5</sup>. JSF is event driven but JSP is not. This imposes severe limitations on mixing JSF and non JSF tag libraries in JSP. The developer must have a clear understanding of JSF and JSP life

<sup>5</sup> <http://onjava.com/pub/a/onjava/2004/06/09/jsf.html#>

cycles. JSF doesn't work very well with REST<sup>6</sup>. JSF security has limited features. It is server dependent and limited to static roles. It cannot support both web and business layer security at the same time<sup>7</sup>.

The unique selling point of the Spring MVC framework was Inversion of Control – IOC. Spring MVC is a simple, lightweight framework. It integrates well with the existing view options – JSP/JSTL, Tiles, Velocity etc. It is flexible and it's very easy to override binding, validation etc. Being too flexible posts another issue – there is no common parent controller. The flexibility is achieved by XML driven configuration and that makes Spring MVC configuration intensive – too much XML. Spring MVC doesn't have AJAX support.

Struts has a very simple architecture and can easily be extended. The whole structure revolves around a controller based or page navigation architecture. It has a good tag library which can easily be customized to work with other frameworks like Velocity. The biggest issue with Struts is the lack of support and documentation. The internet is full of loose discussions which suggest that Struts is soon going to be retired.

In order to do an objective evaluation, we need to define a set of features and how effectively the frameworks support them. In addition to the feature comparison, the resource availability and skills also need to be considered.

Feature	JSF	Struts 2	Spring MVC
AJAX Support	No AJAX support. However JSF 2.0 specifications plan to address AJAX support.  MyFaces or ICEfaces are recommended for AJAX support.	Built in DOJO Support.	No built in support. DWR or third party extras to be used.
Bookmarking and URL	JSF is based on POST. No support for URL.	The namespaces make it easy for handling URLs.	Allows full URL control.
Validation	The default implementation and messages are not neat.	Supports OGNL – Object Graph Navigation language – to support powerful expressions.	Supports 'Commons Validator' framework. This is a mature framework.
Testing	Can be easily tested. Similar to Struts to action testing.	Easy testing with Mocks – Easy Mock, jMock, Spring Mocks	
Ability to Post and redirect	Requires custom solution. Difficult to get the messages to the page bean.	Needs custom solution.	Has built in support to add parameters to redirect.
Internationalization	Supports. One file per locale. Need to declare the resource bundle per page.	Supports. Recommends one file per action / page.	Supports. One file per locale.
Page Decoration	Supports Tiles. However, SiteMesh is not recommended for JSF.	Supports Tiles and SiteMesh.	
Community support	The largest community on the web.	Available	Available
Tools	A large number of tools available on the web.	Available	Available
Resource Availability	Low	High	High
Resource Skills	Beginner	Expert	Expert

<sup>6</sup> <http://matthiaswessendorf.wordpress.com/2008/10/27/rest-and-jsf/>

<sup>7</sup> <http://developers.sun.com/learning/javaonline/2007/pdf/TS-4514.pdf>

The key strength of JSF is in its component based event driven architecture. However, JSF still has to catch up with Struts and Spring MVC on other features. JSF is still a relatively young framework and will need some time before it matures. Since Struts and Spring MVC are around for a long time, the number of skilled resources available is also large. It will take a while before the skilled resource pool for JSF builds.

## 4 JSF Prototype

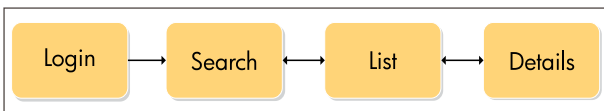
The fastest way to get familiar with a new technology or tool is, usually, to build a quick prototype. The prototype should be carefully chosen so that we don't end up developing a "Hello World" application nor do we end up implementing a full fledged application.

The chosen prototype focuses on the following aspects –

- Capturing data from the screen and populating the backing bean
- Transferring large data sets to the browser – load and performance testing
- Maintaining the state to implement pagination
- User session management
- Event handling – to get the details after clicking on an individual item in the list
- Checking the difference between client side and server side state maintenance

We didn't try AJAX and extensive state management. A shopping cart like application would have helped us to evaluate the state management.

This prototype was developed by a developer who had no knowledge of the JSF technology. It was a read-learn-build attempt. The developer however had experience in Struts and Spring MVC. A similar application was already implemented by the same developer in Struts and Spring MVC. This helped us compare the three frameworks in a more practical way.



Code statistics for the web layer

- 3 JSPs – Search and details were reused
- 2 Backing beans – POJO
- 2 config files
- 8 hours to do a quick reading on JSF
- 4 hours to implement

Feedback from the developer

- JSF has a steeper learning curve compared to Spring MVC and Struts
- The configuration was somewhat difficult to understand

- The mapping was somewhat difficult to understand
- JSF tags are somewhat cumbersome to use and make the page bulky
- Session maintenance is not straightforward
- The internals of JSF – The tree view structure is far too complicated to comprehend

Runtime analysis of the application

- The performance is slower than Spring MVC / Struts
- The size of the HTML generated is substantially bigger than what is generated in Spring MVC or Struts.

## 5 Inputs from the battlefield

The reality has more surprises that one can imagine. No amount of prototype or pilot implementation can match the experience of a full fledged project. Two projects were chosen to do the case study – a fairly large project and another one of more moderate size. The project details are listed below:

	Project 1	Project 2
Duration	24 Months	8 Months
Team Size ( Peak / Avg. )	15 / 10	6/6
JSP Files	750	200
JSF Classes – backing beans and controller	1200	300
Lines of Code – Approx	150K	30K

Detailed discussions with the Architects, Tech Leads and the developers voiced the common and biggest concern around lack of training and expertise in the team. The project experienced the usual delivery timeline pressure. The decision to use JSF was taken without taking into account the ground realities. In the end, all of this reflected in inferior code quality, runtime performance issues and a number of bugs. There was a significant variance between the planned and actual efforts. The immediate temptation was to write off JSF Technology.

The team also faced various technical issues during implementation. The initial version of JSF had some issues which were addressed in the subsequent releases. As a result, some bugs automatically vanished once the new release was used. There is no standard JSF implementation available and multi-flavor implementations are available. In one of the projects, the JSF framework was extended and the team played with the JSF internals. This was not a good idea. Similar functionality could have been implemented in Spring MVC / Struts. The need to extend the JSF framework needs to be critically evaluated, considering the fact that the team didn't have much experience in JSF. JSF

configuration parameters were also not understood clearly by the team. As an example, JSF was configured to maintain state on the client instead of the recommended server side state maintenance. This resulted in huge page size – several MB instead of a few KB. This resulted in lower performance due to large amounts of data transferred between the browser and the web server. JSF tags are a little clumsy to use and they generate a huge HTML page. In one typical case, the size of the page generated was about 1.2 MB and it took about 13 seconds to load in the browser. The same page, when recoded in Struts or Spring MVC was about 200K and loaded in less than a second. In short, JSF is not suitable for high performance applications.

Despite the issues faced with JSF, the teams were surprisingly willing to work with JSF in future projects. The obvious reason was – every one is talking about and using JSF and one doesn't want to miss the boat. There is a lot of talk on the web about JSF.

The main advantage of JSF is the “event programming”. Controlling Widgets offer better control over page as they are more powerful than the standard JSP scripting. The Expression Language<sup>8</sup> (EL) expression helps improve the user interface code readability and maintenance. A combination of JSF and AJAX – MyFaces – helps improving the performance. Ready to use components help reducing the efforts.

## 6 Conclusion

The web is full of conflicting reports, blogs and hypes. We should trust these reports only to an extent. The best way to evaluate and trust a technology is to have the “do it yourself” approach. A quick prototype will give a realistic idea about the technology, the readiness of resources.

In short, the recommendations are as follows:

- Be clear about the application requirements
- Know about the technology. Make sure that you know about JSF.
- Map and match the project requirements and features of JSF. Do you really need all the features of JSF? Can the project be implemented using more light weight technologies?
- Check the available resources, their skill level. Make sure that you have at least one JSF guru in the team.
- JSF provides event handling and state maintenance. Event handling tends to slow down the application and state maintenance consumes a lot of memory. Please make sure that you are aware of these limitations and the project really demands them.
- JSF is a component-based architecture with heavy dependency on state. This has impact on the scalability and functioning in a clustered environment. Client side state maintenance helps in achieving the desired scalability. However, maintaining the state on the client side is an expensive operation – it consumes memory and slows down the application.
- Make sure that you understand the JSF configuration parameters and their implication.
- Please make sure that the resources are skilled and trained. If not, arrange the training.
- Last but certainly not the least, DON'T rush to code.

In case you are not confident or convinced about JSF, just say NO to JSF.

<sup>8</sup> <http://java.sun.com/j2ee/1.4/docs/tutorial/doc/JSPIIntro7.html>

## References:

<http://java.sun.com/javaee/jaserverfaces/>

<http://java.sun.com/javaee/jaserverfaces/download.html>

[http://en.wikipedia.org/wiki/JavaServer\\_Faces#Comparison\\_to\\_other\\_Web-GUI\\_Frameworks](http://en.wikipedia.org/wiki/JavaServer_Faces#Comparison_to_other_Web-GUI_Frameworks)

<http://static.raibledesigns.com/repository/presentations/ComparingJavaWebFrameworks-ApacheConUS2007.pdf>

[http://en.wikipedia.org/wiki/List\\_of\\_web\\_application\\_frameworks#Java](http://en.wikipedia.org/wiki/List_of_web_application_frameworks#Java)

JavaServer Faces in Action, Kito Mann, Manning Publications Co., 2005. ISBN 978-1932394122

JavaServer Faces, Hans Bergsten, O'Reilly Media, 2004. ISBN 978-0596104245

<http://matthiaswessendorf.wordpress.com/2008/10/27/rest-and-jsf/>

## Acknowledgements:

I would like to thank Lakshmi Narasimhan and Ranganaathan for project level inputs. Special thanks to Sumit Kumar Gupta for help in implementation of the prototype.

## Contact us

### USA

Mphasis  
460 Park Avenue South  
Suite # 1101, New York  
NY 10016, U.S.A.  
Tel: +1 212 686 6655  
Fax: +1 212 686 2422

### UK

Mphasis  
88 Wood Street  
London EC2V 7RS, UK  
Tel: +44 208 528 1000  
Fax: +44 208 528 1001

### AUSTRALIA

Mphasis  
410 Concord Road  
Rhodes, NSW 2138, Australia  
Tel: +61 290 221 146  
Fax: +61 290 221 134

### INDIA

Mphasis  
Bagmane Technology Park  
Byrasandra  
C.V. Raman Nagar  
Bangalore 560 093, India  
Tel: +91 80 4042 6000  
Fax: +91 80 2534 6760

## About Mphasis

Mphasis is a global service provider with \$1B in revenues, delivering technology based solutions to clients across the world. With currently over 39,000 people, Mphasis services over 200+ clients in Banking and Capital Markets, Insurance, Manufacturing, Communications, Media & Entertainment, Healthcare & Life Sciences, Transportation & Logistics, Retail & Consumer Packaged Goods, Energy & Utilities, and Governments around the world. Our competency lies in our ability to offer integrated service offerings in Applications, Infrastructure Services & Business Process Outsourcing capabilities. We are uniquely positioned to service our clients with best cost-performance. To know more about Mphasis, log on to [www.mphasis.com](http://www.mphasis.com)