JAVASERVER FACES (JSF)

WHAT IS JSF?
- Is a component oriented and event driven framework.
- Focuses on GUI building and separation of view from controller and model.
- Consists of a set of APIs, implemented using JSP.
- Swing-like component model and GUI design concepts

JSF FEATURES
- Page navigation specification
- Component-centric approach
- Validators & converters
- Java bean management
- Event handling
- Support for visual layout design and rapid application development with IDEs
- Multiple client devices support (not just html)
**COMPONENTS**

- **Basic UI Components**
  - Based on the standard HTML elements.
  - The base class for all JSF components is UIComponent.
  - There are two types of tags:
    - JSF Core Tags
    - JSF HTML Tags
  - To use the tags, have to declare:
    - `<%@ taglib prefix="f" uri="http://java.sun.com/jsf/core"%>`
    - `<%@ taglib prefix="h" uri="http://java.sun.com/jsf/html"%>`

**COMPONENTS CON’T**

- **Custom UI Components**
  - To extend standard components to allow for more advanced UI interaction.
  - Steps to create custom component:
    1. Extend a UIComponent
    2. Define the renderer or implement inline
    3. Create a custom tag that subclasses UIComponentTag

- **3rd Party Component Libraries**
  - Can improve the quality of the web application and save development time.
  - Examples:
    - MyFaces (Tomahawk, Tobago, Trinidad)
    - DojoFaces
    - RC Faces

**COMPONENTS CON’T**

- **JSF Core Tags (starts with f):**
  - Deals with forms and other HTML-specific goodies.
  - For example:
    - `<f:validateLength maximum="40" minimum="6"/>`

- **JSF HTML Tags (starts with h):**
  - Deals with logic, validation, controller, and other tags specific to JSF.
  - For example:
    - `<h:inputText id="meetingTitle" required="true" size="40" value="#{MeetingBean.meetingTitle}"/>`
USER INTERFACE EVENTS

- **Action Controllers**
  - Starts back-end processing
- **Event Listeners**
  - Affects only the format of the user interface
  - Types:
    - ActionListener
      - `<h:commandButton value="..." .../>`
    - ValueChangeListener
      - `<h:selectBooleanCheckbox.../>`

ARCHITECTURE

- **JSF follows Model-View-Controller (MVC) framework.**
  - Improves control flow and communication within the application.
  - Makes it easier to maintain.
- **JSF applies a client-side caching technique in its client components.**
  - Stores data in web pages which the components can access rather than requesting it from the server.
- **JSF structure:**
  - Same as any JSP project, with extra xml configuration files for bean, navigation, and resource specification

TOPICS

- Introduction
- Components and Architecture
- Request Flow Life Cycle
- Ajax in JSF
- JSF vs. Struts
- JSF vs. ASP.Net
- Advantages
- Disadvantages
- Example
- References

REQUEST FLOW LIFE CYCLE

- Follows the same concept of JSP in receiving a request and responding using HTTP.
- But, JSF has more developed request processing flow than JSP.
- JSP has only one phase of execution (render phase) when issuing a request.
The request processing in JSF has more stages.

These processes occur automatically.

To use this life cycle:
- Create the appropriate event handlers in components.
- Define the navigation rules with a few lines of straightforward xml.

The advantages with this complex life cycle:
- Validation rules and other operations like casting and conversion are kept relatively straightforward and concise.
- Events can directly update properties of components in a very object oriented fashion.

The core functionality of Ajax is encapsulated inside the JSF components.

One strategy to implement Ajax in JSF:
1. Embed the Ajax code in custom components.
   ```xml
   <ajaxTags:completionField size="40" id="cityField" completionMethod="#{AutoCompleteTextField.completeCity" value="#{SessionBean.city}" required="true" />
   ```
2. Setups appropriate handlers to accommodate incoming AJAX requests.
3. Ajax code then renders alongside the form elements.
JSF VS. STRUTS

<table>
<thead>
<tr>
<th>JSF</th>
<th>Struts</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Event driven UI Components</td>
<td>• No Events in the OO sense</td>
</tr>
<tr>
<td>• Robust, Reusable and extensible components and tools (Validaters, convertors, etc).</td>
<td>• Support the development of complete applications.</td>
</tr>
<tr>
<td>• Logic is eliminated within JSP files (View)</td>
<td>• Logic remains embedded in the JSP (View)</td>
</tr>
<tr>
<td>• Supports RAD.</td>
<td>• Doesn’t support RAD.</td>
</tr>
<tr>
<td>• Rendering and view are decoupled (Can render to mobile devices and other standards ).</td>
<td>• Code must be rewritten for other standards, where supported.</td>
</tr>
<tr>
<td>• Pushed and supported by major software vendors: Sun, IBM, Oracle.</td>
<td>• Managed, supported and developed as an Apache project.</td>
</tr>
<tr>
<td>• Still relatively young given its complexity. Contains more bugs.</td>
<td>• Well established. Large support base</td>
</tr>
</tbody>
</table>

JSF VS. ASP.NET

- Similarity:
  - They share similar paradigms.
  - Both are built around encapsulating GUI components into virtual objects.
- Difference:
  - ASP .Net is platform dependent (works only on Windows).
  - JSF is platform independent (works with virtually every OS).
**TOPICS**
- Introduction
- Components and Architecture
- Request Flow Life Cycle
- Ajax in JSF
- JSF vs. Struts
- JSF vs. ASP. Net
  - Advantages
  - Disadvantages
- Example
- References

**ADVANTAGES**
- Easy to learn for existing Java web developers.
  - Based on java, expression language similar to JSP
- Follow MVC design pattern.
  - Clean separation between application logic and presentation.
- Enables the use of IDEs for Rapid Application Development (NetBeans, Jdeveloper, Eclipse, etc)
- Extensible and reusable component and rendering architecture.
- Support mark-up Language, protocol, and client device independence.
  - JSF leverages existing, standard UI and web-tier concepts.
- Transparently saves state information and repopulates forms when they redisplay.

**DISADVANTAGES**
- Somewhat immature
  - Initially released in 2004, now on version 1.2, with a major revision (2.0) due in 2008.
  - Not many case studies. Few major projects have been completed with JSF. It’s only just beginning to reach mainstream now.
- Lots of supporting code
  - XML configuration is simpler than struts, but still there.
  - Still more lines of code per task than frameworks such as RoR, Django (this however, also confers some advantages in terms of capabilities and extensibility).
AN EXAMPLE JSF APPLICATION

Files involved
- faces-config.xml
  - Navigation rules, managed beans, resource bundles
- MeetingBean.java
  - Request data is set and retrieved from this managed bean
- MeetingPageBean.java
  - Bound components are handled in this backing bean
- messages.properties
  - Messages and other text are stored in this resource file
- setupmeeting.jsp
- showmeeting.jsp
- JSF core, html and tomahawk libraries

setupmeeting.jsp starts with some JSP to define our JSF tags, and some opening html

```html
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<%@taglib prefix="f" uri="http://java.sun.com/jsf/core"%>
<%@taglib prefix="h" uri="http://java.sun.com/jsf/html"%>
<%@taglib prefix="t" uri="http://myfaces.apache.org/tomahawk"%>

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <link rel="stylesheet" type="text/css" href="basic.css" />
  <title>Meeting Setup</title>
</head>
<body>

Next we start our form (remember the binding="...")
- Then the header, and a grid (table) which contains our labels, inputs and messages

```
```html
</f:view>
```
Different validators required for the various inputs:
- Email Validation
  - `<t:validateEmail message="Invalid Email specified"/>
- Password Length Validation
  - `<f:validateLength maximum="40" minimum="6"/>

The calendar component uses the Apache Tomahawk JSF library.
- `<t:inputCalendar id="meetingDate" monthYearRowClass="yearMonthHeader" weekRowClass="weekHeader" currentDayCellClass="currentDayCell" renderAsPopup="true" value="#{MeetingBean.meetingDate}"/>

Hide and Show buttons
- Placed in a separate form so they don’t disappear too.
  - `<h:form id="hideForm">
    - `<h:commandButton id="hide" action="#{MeetingPageBean.hideForm}" value="Hide Form"/>
    - `<h:commandButton id="show" action="#{MeetingPageBean.showForm}" value="Show Form"/>
  `</h:form>

Beans
- `<managed-bean>
  - `<managed-bean-name>MeetingBean</managed-bean-name>
  - `<managed-bean-class>c410proj.MeetingBean</managed-bean-class>
  - `<managed-bean-scope>request</managed-bean-scope>
- </managed-bean>

Actions
- `<navigation-rule>
  - `<from-view-id>*</from-view-id>
  - `<navigation-case>
    - `<from-outcome>setMeeting</from-outcome>
    - `<to-view-id>/showmeeting.jsp</to-view-id>
- </navigation-case>
- </navigation-rule>

Resource-bundles
- `<resource-bundle>
  - `<base-name>myPackage.messages</base-name>
  - `<var>msgs</var>
- </resource-bundle>

messages.properties
- formHeading=Please enter the following
- invalidEmailMsg=Invalid Email specified
- javax.faces.component.UIInput.REQUIRED=Field is required
  - Overrides ‘required’ tag default message

MeetingBean.java
- public class MeetingBean {
  - private String organizerName;
  - private String organizerEmail;
  - private String password;
  - private String meetingTitle;
  - private Date meetingDate;
  - public MeetingBean() {
      organizerName = new String();
      ...}
  - public String getOrganizerName() { return organizerName; }
  - public void setOrganizerName(String name) { this.organizerName = name; }
  - ...}
AN EXAMPLE JSF APPLICATION

- Linked to the meetingForm specified in setupmeeting.jsp
  - `<h:form id="meetingForm" binding="#{MeetingPageBean.form}"/>

MeetingPageBean.java

```java
public class MeetingPageBean {
    private UIForm form;

    public UIForm getForm() { return form; }
    public void setForm(UIForm form) { this.form = form; }
    public void hideForm() { form.setRendered(false); }
    public void showForm() { form.setRendered(true); }
}
```

showmeeting.jsp

```html
<f:view>
    <h2><h:outputText value="Meeting Created: #{MeetingBean.meetingTitle}"/></h2>
    <h:panelGrid columns="2">
        <f:facet name="header">
            <h:outputText value="Meeting Options"/>
        </f:facet>
        <h:outputText value="Meeting Title"/>#{MeetingBean.meetingTitle}"
        <h:outputText value="Organizer Name"/>#{MeetingBean.organizerName}"
        <h:outputText value="Organizer Email"/>#{MeetingBean.organizerEmail}"
        <h:outputText value="Meeting Date"/>#{MeetingBean.meetingDate}"
    </h:panelGrid>
</f:view>
```

TOPICS

- Introduction
- Components and Architecture
- Request Flow Life Cycle
- Ajax in JSF
- JSF vs. Struts
- JSF vs. ASP. Net
- Advantages
- Disadvantages
- Example
- References

REFERENCES


![Image](http://files.wordpress.com/2012/02/)