JSF

Java Server Faces

- Is in some ways built upon STRUTS
- Is used instead of STRUTS
- Both JSF and STRUTS are APIs for webinterfaces and the interface layer.

JSF

- Idea comes from .NETs WebForm
- Also a way to get a common (among IDEs) graphical webinterface builder
- A way to minimize M$s advantage with homogeneous environment

- Even though HTTP lacks state, you get the feeling you get state in JSF
- It function almost like Swing.
- You can say it act on component level rather than JSP level.

- Every page contain components like
  - Forms
  - input
  - buttons
- Every time the user do something you get an event call on the server.
- A Controller is called,the servlet javax.faces.webapp.FacesServlet

- FaceServlet creates event calls to the event listeners.
- You point out the Servlet call in a xml file.
-

i web.xml

```
<!-- Faces Servlet -->
<servlet>
    <servlet-name>Faces Servlet</servlet-name>
    <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
</servlet>
<!-- Faces Servlet Mapping -->
<servlet-mapping>
    <servlet-name>Faces Servlet</servlet-name>
    <url-pattern>*.jsf</url-pattern>
</servlet-mapping>
```

- Write a jsp file using JSF components.
- Write a Java Bean which contain the component data and handle input.
- Write an EventListener

JSP fil

```
<%@ taglib uri="http://java.sun.com/jsf/html" prefix="h" %>
  <%@ taglib uri="http://java.sun.com/jsf/core" prefix="f" %>
  <body bgcolor="white">
  <f:view>
  <h:form id="helloform">
  <h2>What is your name?</h2>
        <h:inputText id="username" value=”#{HelloBean.userName}"/>
        <h:commandButton id="submit" action="success" value="Submit"/>
  </h:form>
  </f:view>
```

- h:view is central, it defines a view which is the top tag for all JSF components
- h:form is a input form.
- h:outputText is a label.
- h:inputText is a text input field.
- h:commandButton adds a button.

Navigation rules

<navigation-rule>

<from-view-id>/hello.jsf</from-view-id>

<navigation-case>

<from-outcome>sucess</from-outcome>

<to-view-id>/helloNext.jsf</to-view-id>

</navigation-case>

</navigation-rule>

---- inuti faces-config.xml

Böna kopplad till komponenter

```
package hello;

import javax.faces.component.UIComponent;

public class HelloBean {String str = null;

public void setUserName(String name) {

      str = name; }

public String getUserName() {

      return str; }

public String getResponse() {

      return str;

}
```

faces-config.xml

```
<managed-bean>
<managed-bean-name>HelloBean</managed-bean-name>
<managed-bean-class>hello.HelloBean</managed-bean-class>
<managed-bean-scope>session</managed-bean-scope>
</managed-bean>
```

dataTable

- Används ungefär som en table när man vill ha en uppräkning av objekt.
- Du får en tabell som visar på en mängd objekt av en typ på ett dynamiskt sätt.

- <h:dataTable id="dt1" value= "#{TableBean.all}" var="item">
- <f:facet name="header">
- <h:outputText value="Fin header" />
-

-
- <h:column>
- 	<f:facet name="header">
- 	<h:outputText value="id" />
- 	</f:facet>
- 		<h:outputText value="#{item.id}"></h:outputText>
- </h:column>

- <h:column>
- <f:facet name="header">
- <h:outputText value="name"/>
- </f:facet>
- <h:outputText value="#{item.name}"></h:outputText>
- </h:column>

- </f:facet>
- <f:facet name="footer">
- 	<h:outputText value="Foten!" />
- </f:facet>
-
- </h:dataTable>