

FACELETS PAGE LAYOUT 2.0

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
  Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
  xmlns:f="http://java.sun.com/jsf/core"
  xmlns:h="http://java.sun.com/jsf/html"
  xmlns:ui="http://java.sun.com/jsf/facelets">
<h:head>...</h:head>
<h:body>
  <h:form>
    ...
  </h:form>
</h:body>
</html>
```

TEXT FIELD

12345678901234567890

page.xhtml

```
<h:inputText value="#{bean1.luckyNumber}">
```

WEB-INF/classes/com/corejsf/SampleBean.java

```
@Named("bean1") // or @ManagedBean(name="bean1") 2.0
@SessionScoped
public class SampleBean {
  public int getLuckyNumber() { ... }
  public void setLuckyNumber(int value) { ... }
  ...
}
```

BUTTON

press me

page.xhtml

```
<h:commandButton value="press me" action="#{bean1.login}"/>
```

WEB-INF/classes/com/corejsf/SampleBean.java

```
public class SampleBean {
  public String login() {
    if (...) return "success"; else return "error";
  }
  ...
}
```

The outcomes success and error can be mapped to pages in faces-config.xml. If no mapping is specified, the page /success.xhtml or /error.xhtml is displayed. 2.0

GET REQUESTS 2.0

```
<f:metadata>
  <f:viewParam name="item" value="#{bean1.currentItem}"/>
  <f:viewParam name="userId" value="#{bean1.user}"/>
</f:metadata>
```

Request parameters set bean properties before the page is rendered.

```
<h:button value="Continue" outcome="#{bean1.continueOutcome}"
  includeViewParams="true"/>
```

The getContinueOutcome method is called *when the button is rendered*. The view parameters are added to the request URL.

RADIO BUTTONS

Cheese Pickle Mustard Lettuce Onions

page.xhtml

```
<h:selectOneRadio value="#{bean1.condiment}>
  <f:selectItems value="#{bean1.choices}" var="it" 2.0
    itemLabel="#{it.description}"
    itemValue="#{it.productId}"/>
</h:selectOneRadio>
```

WEB-INF/classes/com/corejsf/SampleBean.java

```
public class SampleBean {
  public Collection<Condiment> getChoices() { ... }
  public int getCondiment() { ... }
  public void setCondiment(int value) { ... }
  ...
}
```

WEB-INF/classes/com/corejsf/Condiment.java

```
public class Condiment {
  public String getDescription() { ... }
  public int getProductId() { ... }
}
```

CONVERSION

```
<h:outputText value="#{bean1.amount}>
  <f:convertNumber type="currency"/>
</h:outputText>
```

The number is displayed with currency symbol and group separator: \$1,000.00

VALIDATION

Using the bean validation framework (JSR 303) 2.0

```
public class SampleBean {
  @Max(1000) private BigDecimal amount;
}
```

Page-level validation and conversion

```
<h:inputText value="#{bean1.amount}" required="true">
  <f:validateDoubleRange maximum="1000"/>
</h:inputText>
```

Error messages

Amount Amount: 'too much' is not a number.
Example: 99

```
Amount
<h:inputText id="amt" label="Amount" value="#{bean1.amount}"/>
<h:message for="amt"/>
```

RESOURCES

page.xhtml 2.0

```
<h:outputStylesheet library="css" name="styles.css"/>
...
<h:message for="amt" errorClass="errors">
```

resources/css/styles.css

```
.errors {
  font-style: italic;
  color: red;
}
```

MESSAGE BUNDLES

faces-config.xml

```
<application>
  <resource-bundle>
    <base-name>com.corejsf.messages</base-name>
    <var>msgs</var>
  </resource-bundle>
</application>
```

WEB-INF/classes/com/corejsf/messages.properties

```
goodbye=Goodbye
```

WEB-INF/classes/com/corejsf/messages_de.properties

```
goodbye=Auf Wiedersehen
```

page.xhtml

```
<h:outputText value="#{msgs.goodbye}!" styleClass="greeting">
```

TABLE WITH LINKS

Name	
Washington, George	Delete
Jefferson, Thomas	Delete
Lincoln, Abraham	Delete
Roosevelt, Theodore	Delete

page.xhtml

```
<h:dataTable value="#{sampleBean.entries}" var="row"
  rowClasses="even,odd">
  <h:column>
    <f:facet name="header">Name</f:facet>
    <h:outputText value="#{row.name}"/>
  </h:column>
  <h:column>
    <h:commandLink value="Delete" immediate="true"
      action="#{bean1.deleteAction(row.id)}" /> 2.0
  </h:column>
</h:dataTable>
```

WEB-INF/classes/com/corejsf/SampleBean.java

```
public class SampleBean {
  public String deleteAction(int idToDelete) {
    // delete the entry whose id is idToDelete
    return null;
  }
  public List<Entry> getEntries() { ... }
  ...
}
```

WEB-INF/classes/com/corejsf/Entry.java

```
public class Entry {
  public int getId() { ... }
  public String getName() { ... }
}
```

AJAX 2.0

```
<h:commandButton value="Update">
  <f:ajax execute="input1 input2" render="output1"/>
</h:commandButton>
```

When the button is clicked, run the phases from “Restore View” until “Invoke Application” on the components with IDs input1 input2, and the “Render Response” phase on the component with ID output1.

Ajax actions are triggered by onclick events for buttons and links, onchange events for input fields and selection lists.

HEADINGS AND SIDEBARS 2.0

templates/masterLayout.xhtml

```
<h:body>
  <div id="heading">
    <ui:insert name="heading">
      <ui:include src="/sections/defaultHeading.xhtml1"/>
    </ui:insert>
  </div>
  <div id="sidebarLeft">
    <ui:insert name="sidebar">
      <ui:include src="/sections/defaultSidebar.xhtml1"/>
    </ui:insert>
  </div>
  <div id="content">
    <ui:insert name="content"/>
  </div>
</h:body>
```

Use CSS to lay out the heading, sidebar, and content.

page.xhtml

```
<!-- Everything outside ui:composition is ignored -->
<ui:composition template="/templates/masterLayout.xhtml">
  <ui:define name="heading">
    <ui:include src="/sections/pageHeading.xhtml1"/>
  </ui:define>
  <ui:define name="content">
    <!-- Place page content here -->
  </ui:define>
</ui:composition>
```

COMPOSITE COMPONENTS 2.0

```
<html xmlns="http://www.w3.org/1999/xhtml"
  xmlns:h="http://java.sun.com/jsf/html"
  xmlns:composite="http://java.sun.com/jsf/composite">
  <composite:interface>
    <composite:attribute name="image" required="true" />
    <composite:attribute name="actionMethod"
      method-signature="java.lang.String action()" />
  </composite:interface>
  <composite:implementation>
    <h:commandLink action="#{cc.attrs.actionMethod}">
      <h:graphicImage url="#{cc.attrs.image}"/>
    </h:commandLink>
  </composite:implementation>
</html>
```

Use the component like this:

```
<util:icon image="#{resource['images/help.png']}"
  actionMethod="#{bean1.showHelp}"/>
```