

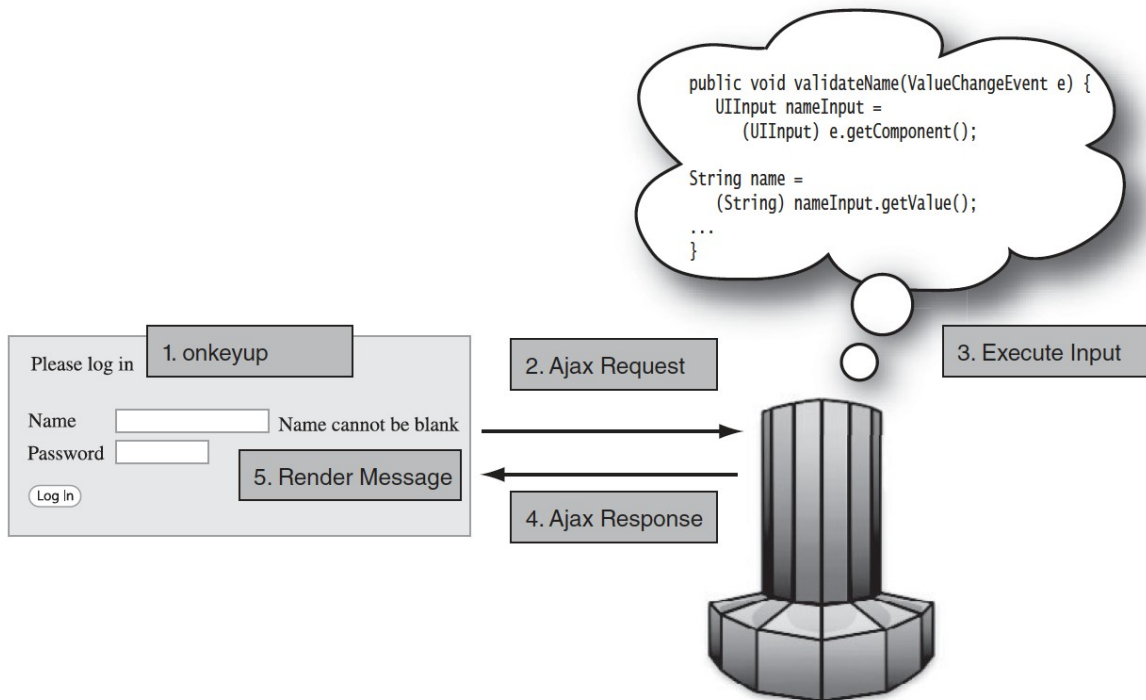
# ASDV 2620 Web App Dev II

## Ajax I

### Ajax and JSF

Ajax requests differ from regular HTTP requests in only two ways: 1. Ajax partially processes forms on the server during the Ajax call. 2. Ajax partially renders Document Object Model (DOM) elements on the client after the Ajax call returns from the server.

This sequence of events is illustrated in Figure below which illustrates an Ajax call that validates a single input, presumably when the field loses focus.



1. JSF Ajax requests partially process components on the server
2. JSF Ajax partially renders components on the client when the request returns.

### The Ajax life cycle has EXECUTE and RENDER cycles

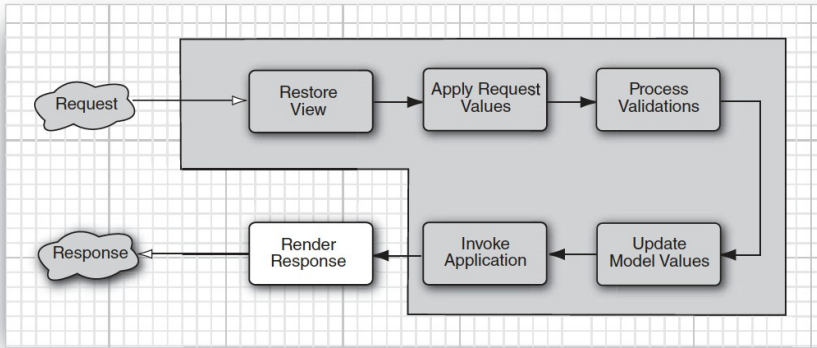
On any given Ajax request, you specify a set of components that JSF executes, and another set of components that it renders.

#### EXECUTE Cycle

The Ajax Life cycle for execute ( where components are executed)

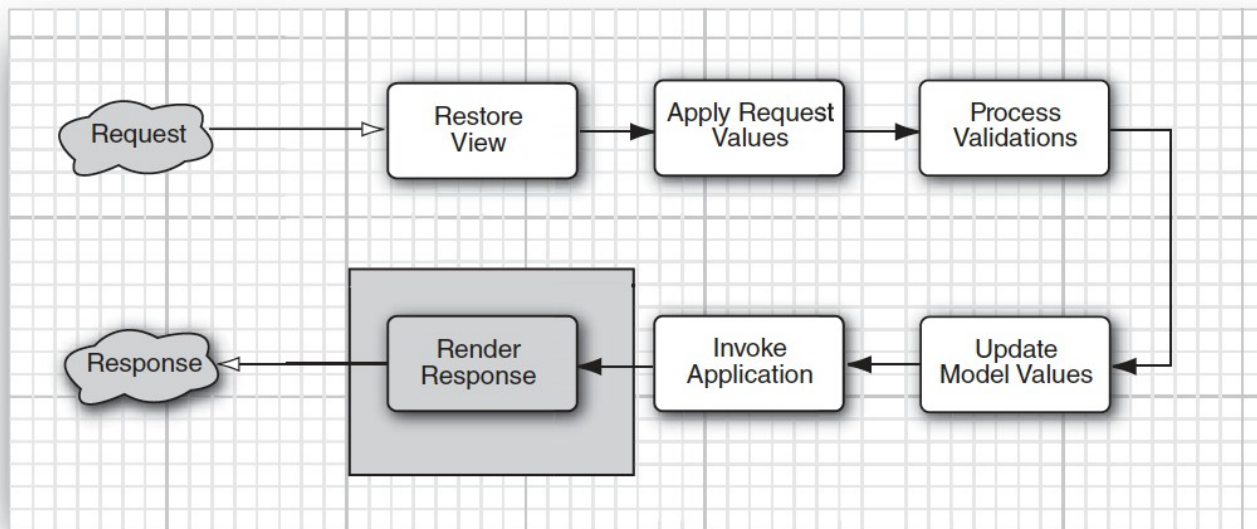
The execute part of the life cycle executes inputs on the server side.

1. Converts and validates the component's value (if the component is an input)
2. Pushes valid input values to the model (if the component is wired to a bean property)
3. Executes actions and action listeners (if the component is an action)



#### The Ajax life cycle for render

The render part of the life cycle, renders components on the client side. For regular *HTTP requests*, all components in a form are both executed and rendered, whereas for *Ajax requests*, JSF executes one or more components, and renders zero or more components.



## The JSF Ajax Execute and Render

1. Associate a component and an event with an Ajax request.
2. Identify components to execute on the server during the Ajax request.
3. Identify components to render after the Ajax request.

### Example

1. Associate an Ajax call with an event, such as *keyup* or *blur*, fired by a specific component.
2. specify the components that you want to execute, and the components you want to render.

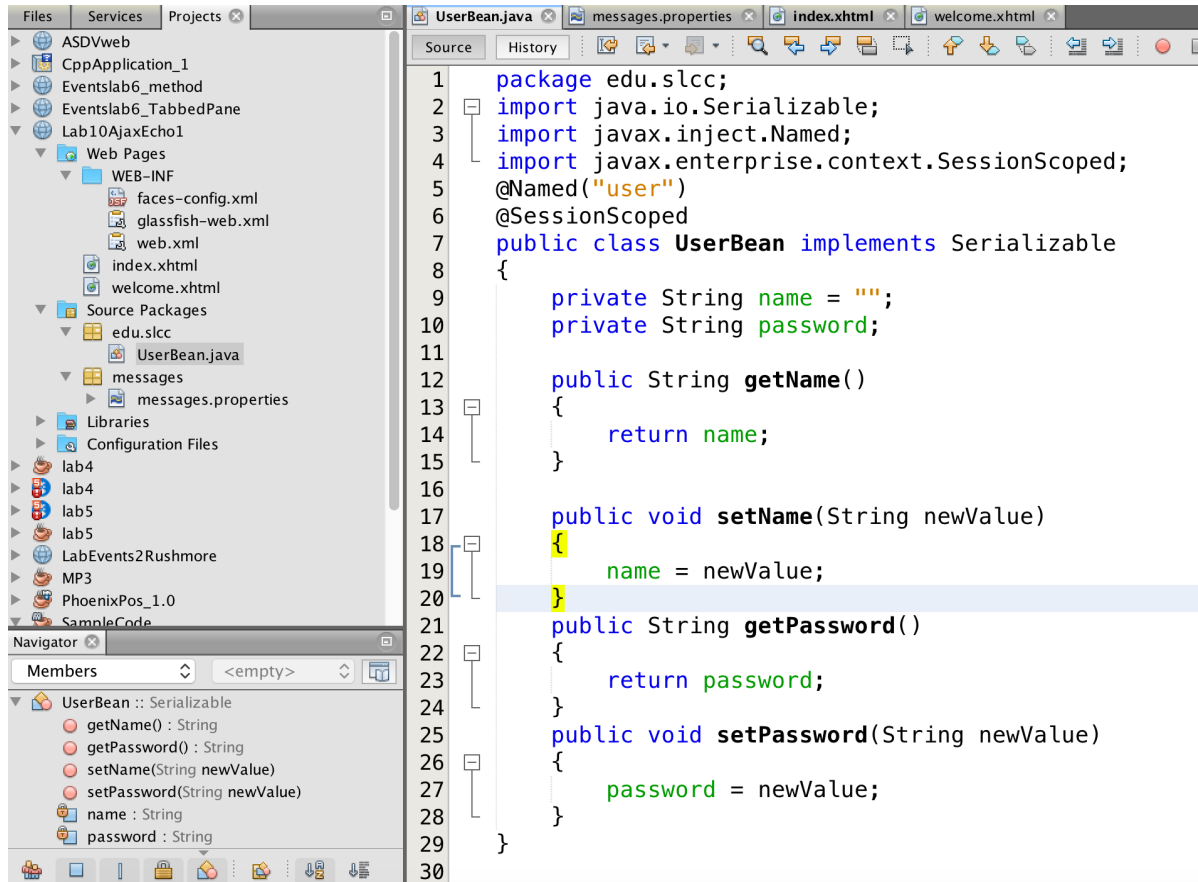
```
<h:inputText id="name" value="#{user.name}">  
  <f:ajax event="blur" execute="@this" render="nameError"/>  
</h:inputText>
```

Here, the code triggers an Ajax event when the input loses focus. That Ajax request executes the **"name"** component on the server—the **@this** value for the `execute` attribute refers to the `f:ajax` tag's surrounding input.

Then, renders a component whose id is **nameError** on the client, when the Ajax call returns.

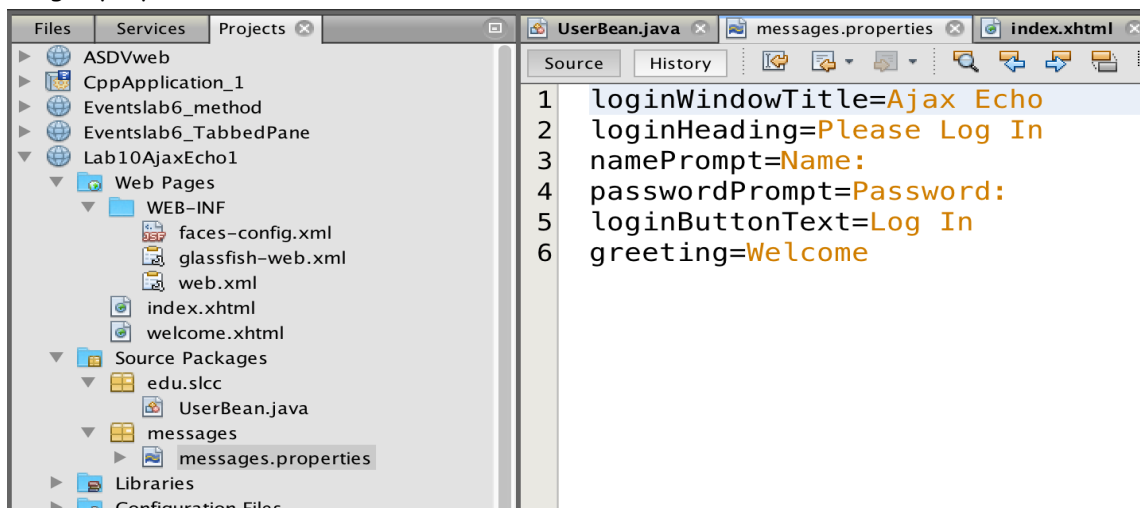
Let us check out the Ajax tag and write a program that echos every character the user types.

1. Copy and paste an older project that has messages, properties and the configuration file. Then refactor it into **LabAjax1Echo**.
2. Create the **UserBean** shown below with setters and getters



```
1 package edu.slcc;
2 import java.io.Serializable;
3 import javax.inject.Named;
4 import javax.enterprise.context.SessionScoped;
5 @Named("user")
6 @SessionScoped
7 public class UserBean implements Serializable
8 {
9     private String name = "";
10    private String password;
11
12    public String getName()
13    {
14        return name;
15    }
16
17    public void setName(String newValue)
18    {
19        name = newValue;
20    }
21
22    public String getPassword()
23    {
24        return password;
25    }
26    public void setPassword(String newValue)
27    {
28        password = newValue;
29    }
30 }
```

The messages.properties file:



```
1 loginWindowTitle=Ajax Echo
2 loginHeading=Please Log In
3 namePrompt=Name:
4 passwordPrompt=Password:
5 loginButtonText=Log In
6 greeting=Welcome
```

The `index.xhtml JSF` file: Observe the the `id="echo"` at line 17 is used also at line 29 to echo. Without the id echo, no echo will occur.

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
3 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4 <html xmlns="http://www.w3.org/1999/xhtml"
5 xmlns:h="http://java.sun.com/jsf/html"
6 xmlns:f="http://java.sun.com/jsf/core">
7 <h:head>
8 <title>#{msgs.loginWindowTitle}</title>
9 </h:head>
10
11 <h:body style="background: skyblue">
12 <h:form>
13 <h3>#{msgs.loginHeading}</h3>
14 <h:panelGrid columns="2">
15 <h:inputText id="inputName" value="#{user.name}"
16 <f:ajax event="keyup" execute="@this" render="echo"/>
17 </h:inputText>
18
19 <h:inputSecret id="inputPassword" value="#{user.password}" size="8"/>
20
21 <h:commandButton value="#{msgs.loginButtonText}" action="welcome"/>
22 </h:panelGrid>
23
24 <p><h:outputText id="echo" value="#{user.name}"/></p>
25 </h:form>
26 </h:body>
27 </html>

```

### The faces-config.xml

```

1 <?xml version="1.0"?>
2 <faces-config xmlns="http://java.sun.com/xml/ns/javaee"
3 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4 xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
5 http://java.sun.com/xml/ns/javaee/web-facesconfig_2_0.xsd"
6 version="2.0">
7 <application>
8 <resource-bundle>
9 <base-name>messages.messages</base-name>
10 <var>msgs</var>
11 </resource-bundle>
12 </application>
13 </faces-config>

```

### The welcome.xhtml JSF

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
3 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
4 <html xmlns="http://www.w3.org/1999/xhtml"
5 xmlns:h="http://java.sun.com/jsf/html">
6 <h:head>
7 <title>#{msgs.loginWindowTitle}</title>
8 </h:head>
9
10 <h:body style="background: skyblue">
11 <h3>#{msgs.greeting} #{user.name}</h3>
12 </h:body>
13 </html>

```

**Table 10–1 f:ajax Tag Attributes**

Attribute	Description
disabled	Disable the tag by specifying true for the disabled attribute.
event	<p>The event that triggers the Ajax request. Event names can be JavaScript event names without the on prefix. For example, you would use event="blur" for the onblur event.</p> <p>Event names can also be the following component events: action and valueChange. Those names can be specified for command components (buttons and links) and inputs, respectively.</p>
execute	<p>A space-separated list of components that JSF executes on the server during the Ajax call. Valid keywords for the execute attribute are:</p> <p>@this @form @all @none</p> <p>If you don't specify an execute attribute, JSF uses @this as the default value.</p>
immediate	If you set this attribute to true, JSF processes inputs early in the life cycle.
onerror	A JavaScript function that JSF calls if the Ajax call results in an error.
onevent	<p>A JavaScript function that JSF calls for Ajax events. This function will be called three times during the lifetime of a successful Ajax call:</p> <p>begin complete success</p> <p>For a successful Ajax call, JSF invokes the onevent function when the Ajax call begins (begin), when it has been processed on the server (complete), and just before JSF renders the Ajax response (success).</p> <p>If there is an error during an Ajax request, JSF calls the onevent function after the Ajax request completes, and subsequently invokes the error handler referenced by the onevent attribute.</p>

**Table 10–1 f:ajax Tag Attributes (cont.)**

Attribute	Description
listener	<p>JSF invokes this listener’s processAjaxBehavior method once during each Ajax call, in the Invoke Application phase of the life cycle (at the end of the execute portion of the life cycle).</p> <p>That method must have this signature: <code>public void processAjaxBehavior(javax.faces.event.AjaxBehaviorEvent event)</code> throws <code>javax.faces.event.AbortProcessingException</code></p>
render	<p>A space-separated list of components that JSF renders on the client after the Ajax call returns from the server.</p> <p>You can use the same keywords (<code>@all</code>, <code>@this</code>, <code>@form</code>, and <code>@none</code>) that are valid for the execute attribute.</p> <p>If you do not specify the render attribute, it defaults to <code>@none</code>, meaning JSF will not render any components after the Ajax request completes.</p>

**Naming convention JSF uses for events**

Take the JavaScript event name, and strip the leading on. So **onblur becomes blur**, **onkeyup becomes keyup**, etc.

Events can be the component events `action` and `valueChange`, instead of JavaScript events.

The `onerror` and `onevent` attributes are JavaScript functions that JSF calls when certain predetermined events happen in the Ajax life cycle.

For a successful Ajax request, JSF invokes the **onevent** function three times: when the Ajax request begins, when it completes, and again after completion,.

JSF invokes the **onerror** JavaScript function after an unsuccessful Ajax request. The value for the listener attribute is a method expression. JSF calls that Java method once per Ajax call (in the Invoke Application phase of the JSF life cycle).