Faces Flow in JSF 2.2 – Part 1: Basics

Originals of Slides and Source Code for Examples:
http://www.coreservlets.com/JSF-Tutorial/jsf2/

For live training on JSF 2, PrimeFaces, or other Java EE topics, email hall@coreservlets.com
Marty is also available for consulting and development support

Taught by the author of *Core Servlets and JSP*, this tutorial, and JSF 2.2 version of *Core JSF*. Available at public venues, or customized versions can be held on-site at your organization.

- Courses developed and taught by Marty Hall
  – JSF 2, PrimeFaces, Ajax, jQuery, Spring MVC, JSP, Android, general Java, Java 8 lambdas/streams, GWT, custom topic mix
  – Courses available in any location worldwide. Maryland/DC area companies can also choose afternoon/evening courses.
- Courses developed and taught by coreservlets.com experts (edited by Marty)
  – Hadoop, Spring, Hibernate/JPA, RESTful Web Services

Contact hall@coreservlets.com for details

Topics in This Section

- **Motivation**
- **Setup**
  - Integrating Glassfish 4 within Eclipse
  - Using Faces Flow and CDI with Tomcat
- **Implicit navigation with faces flow conventions**
- **Flow-scoped beans and the flowScope EL variable**
- **Configuring flows with flow-specific XML files**
- **Configuring flows with the global XML file**

Overview

Big Ideas

- **Flows are groups of pages**
  – Can enter flow at only a single place (start page)
  – Once inside, can navigate among any pages in flow
  – Can leave flow to one or more predefined outside pages
  – Can also have nested flows
- **Flows use flow-scoped beans**
  – Wider scope than view scope: bean lives for same user in all pages in flow
  – Narrower scope than session scope: when user navigates outside flow, bean is removed from session and made available for garbage collection
  – There is also a new flowScope EL variable

Motivation

- **Modularity**
  – Pages and beans can self contained
- **Wizard-like structures**
  – Fits well with a common approach to many Web activities. Similarly, the Oracle Java EE tutorial says that it is analogous to a subroutine, with
    - A well defined entry point, list of parameters, and return value.
    - A scope, allowing information to be available only during the invocation of the flow and to not consume resources once the flow returns.
    - The ability to call other flows before returning.
- **Less server memory**
  – Compared to session scope, beans are removed from memory much sooner: when user leaves the flow.
- **Not fooled by multiple browser tabs or windows**
  – Each tab or window is considered to be a separate flow. This is in contrast to session scope, which is easily confused by multiple tabs or windows.

Setup

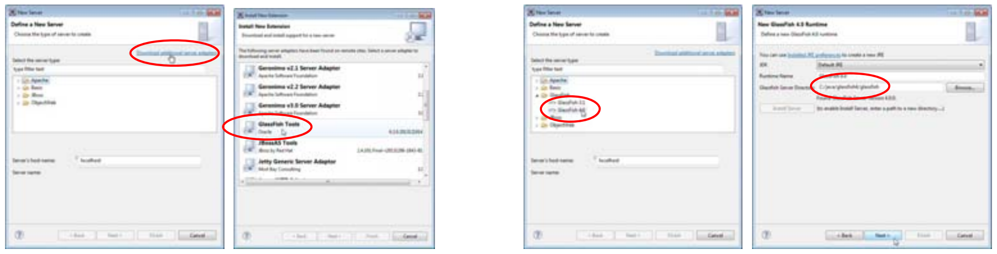
- **Eclipse users**
  - Declare project as Dynamic Web Project for Glassfish 4
    - Integrate Glassfish 4 inside Eclipse (see next slide)
- **Users of other IDEs**
  - Declare project as Java EE 7 Web project (details vary)
- **Faces Flow uses CDI**
  - So, you need a Java EE 7 server, or you must add CDI to a servlet engine like Tomcat.
    - See upcoming slides on adding CDI to Tomcat
- **Glassfish requires an empty beans.xml file**
  - Needs only valid start and end tags, but no body.
    - Java-based configuration fails without the file.

Options for Using Faces Flow

- **Background**
  - Most JSF 2.2 apps can run on Tomcat or Jetty or another servlet container that is free and starts and stops quickly
    - Just need the JSF JAR file added to the app
  - But, faces flow depends on CDI, so needs more
- **Options for apps that use faces flow**
  - A full Java EE 7 server
    - Glassfish 4, Jetty Java EE 7 Web Profile, Firefly (aka JBoss) 8, WebLogic 12, WebSphere 9, TMax Jeus, Geronimo 2, etc.
    - These run faces flow apps with no configuration needed
  - A servlet container with CDI added
    - Tomcat, Jetty, Resin, etc.
    - You must add the CDI JAR file and make additional server configurations. Details at http://www.jsf2.com/using-cdi-and-jsf-2.2-faces-flow-in-tomcat/

Integrating Glassfish inside Eclipse

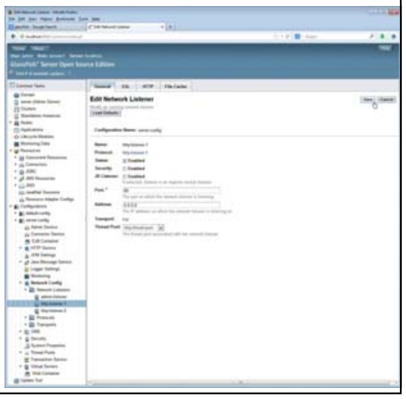
- **Summary**
  - Install Glassfish and Glassfish adapter, add Glassfish server
- **Details**
  - Download from https://glassfish.java.net/download.html
  - R-click in Server tab, New → Server,
  - Click “Download additional server adapters” at top right.



  - After installing Glassfish adapter:
    - R-click, New → Server → Glassfish → Glassfish 4.0.
    - Point at wherever you installed Glassfish

Changing Glassfish Port

- **Using port 80 is convenient**
  - Lets you type http://localhost/blah instead of http://localhost:8080/blah
- **Option 1**
  - Edit <network-listener> port in install_dir\glassfish\domains\domain1\config\domain.xml
- **Option 2**
  - Start Glassfish. Connect to the admin interface (by default on port 4848)
  - In the left menu go to Configurations
  - Select server-config
  - Go to Network Config
  - Go to Network Listeners
  - Click on the appropriate listener, usually http-listener-1
  - Change the Port value to 80. Click Save at top.
  - Restart Glassfish



Using CDI in Tomcat

- **Very brief summary**
  - Add <Resource> to bottom of Tomcat's context.xml
  - Put weld-servlet.jar in WEB-INF/lib of your app
  - Add <resource-env-ref> to web.xml of your app
- **Full details and sample files**
  - http://www.jsf2.com/using-cdi-and-jsf-2.2-faces-flow-in-tomcat/
- **Notes**
  - The exact same app will *not* run unchanged in both Tomcat and a Java EE 7 server, since apps with weld-servlet.jar will not run in Glassfish 4.
    - However, all of the faces flow code is identical, and both Glassfish and Tomcat versions of the app for this tutorial section can be downloaded from the JSF 2 tutorial at http://www.coreservlets.com/JSF-Tutorial/jsf2/

Implicit Navigation with Faces Flow Conventions

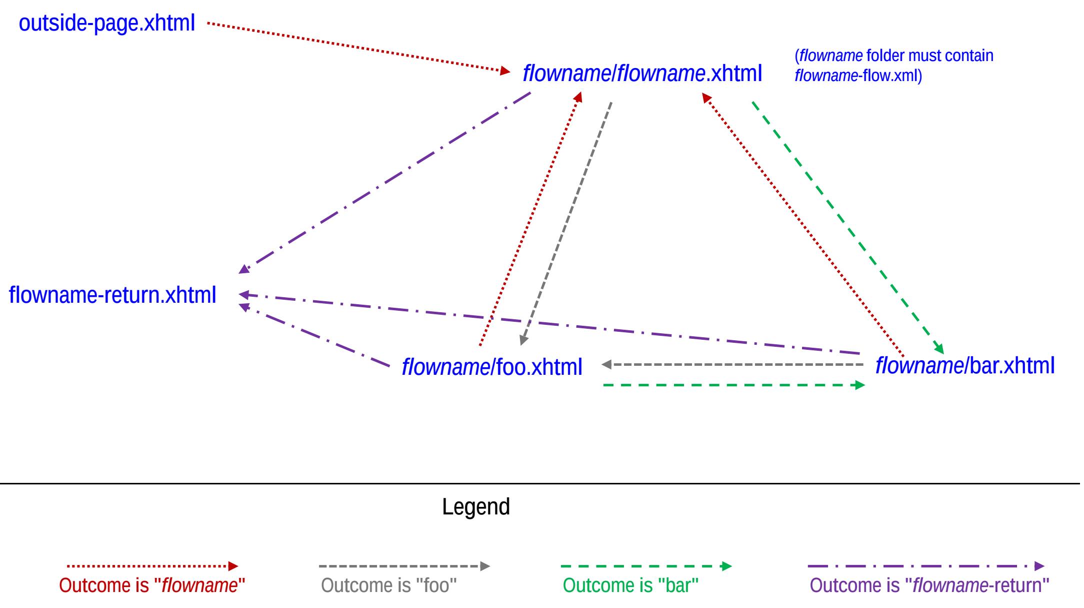
Main Points

- **You use folder that matches flow name**
  – yourFlowName/
- **Folder contains empty XML file**
  – yourFlowName/yourFlowName-flow.xml
- **Default start page matches flow name**
  – yourFlowName/yourFlowName.xhtml
- **Other pages in flow have arbitrary names**
  – yourFlowName/anything.xhtml
  – These are accessible only from within flow
- **Exit (return) page uses flow name**
  – yourFlowName-return.xhtml
    - This is *not* in the yourFlowName folder

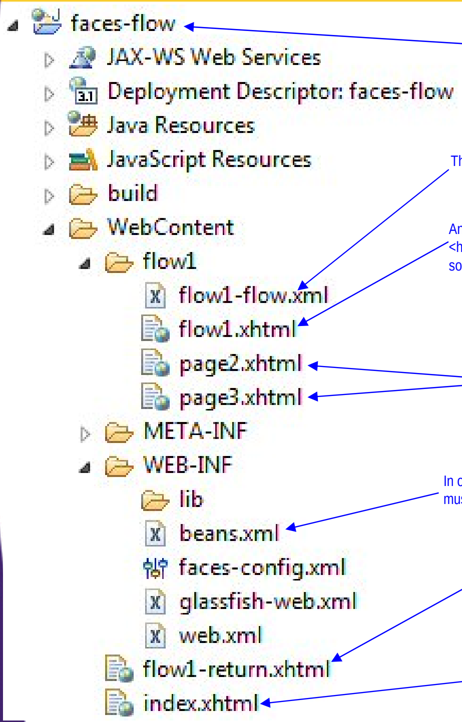
Details

- **Standard files and corresponding outcomes**
  – Folder name matching flow name with empty XML file
    - someFlow/someFlow-flow.xml
  – A default starting page in the folder
    - someFlow/someFlow.xhtml
    - Outcome of "someFlow" enters flow and starts at this page
    - If in flow, outcome of "someFlow" goes to this page
  – Any other files in the folder
    - someFlow/other.xhtml
    - If in flow, outcome of "other" goes to this page.
    - Not reachable from outside the flow.
  – A return page outside the folder
    - someFlow-return.xhtml
    - Outcome of "someFlow-return" exits flow and goes to this page

Implicit Navigation: Flow Diagram



Implicit Navigation Example: Annotated Project Layout



```
faces-flow
  JAX-WS Web Services
  Deployment Descriptor: faces-flow
  Java Resources
  JavaScript Resources
  build
  WebContent
    flow1
      flow1-flow.xml
      flow1.xhtml
      page2.xhtml
      page3.xhtml
    META-INF
    WEB-INF
      lib
      beans.xml
      faces-config.xml
      glassfish-web.xml
      web.xml
    flow1-return.xhtml
    index.xhtml
```

When the project was created, it was designated as a Dynamic Web Project with a target runtime of Glassfish 4.0 and a configuration for Glassfish with JSF 2.2. For non-Eclipse-users, this just means it is a Java EE 7 Web project.

This file must exist, but can be totally empty. Not even start or end tags are needed.

An outcome of "flow1" causes the flow to start, and this page to be displayed. For example, <h:commandButton … action="flow1"/> or <h:commandButton … action="#{someBean.someMethod}"/>, where someMethod returns "flow1".

From within the flow, outcomes of "page2" and "page3" invoke these pages.
The pages are not accessible from outside the flow: you must begin flows at the start page.

In order for Java-based flow configuration (in Faces Flow Part 2) to work on Glassfish 4, this file must exist, and must have valid start and end tags. No content inside the tags is needed.

From within the flow, an outcome of "flow1-return" invokes this page. That outcome also results in the flowScope variable being cleared and flow-scoped beans being made inaccessible and available for garbage collection. (The flowScope variable and flow-scoped beans are discussed in the next section.)

This page cannot be reached from within the flow by using an outcome. Of course, a normal hypertext link will still work. And, we will see that when you use an XML flow-configuration file, you can designate as many return pages as you want.

Reminder: Static Navigation

- **Usual case: run Java code returning String**
  - <h:commandButton action="#{bean.doNav}" .../>
  - <h:commandLink action="#{bean.doNav}" .../>
- **Suppose doNav is very simple**

```
public String doNav() {
  return("result-page");
}
```

- **Equivalent case: specify outcome directly**
  - <h:commandButton action="result-page" .../>
  - <h:commandLink action="result-page" .../>

Flow1 Example: Starting the Flow

- **Link from home page**

```
<h:form>
  <h:commandLink value="Flow 1." action="flow1"/>
</h:form>
```

- **flow1/flow1.xhtml**

```
<ui:composition xmlns="http://www.w3.org/1999/xhtml"
    xmlns:ui="http://xmlns.jcp.org/jsf/facelets"
    template="/templates/flow1-template.xhtml">

  <ui:define name="title">Flow 1 Start Page</ui:define>

</ui:composition>
```

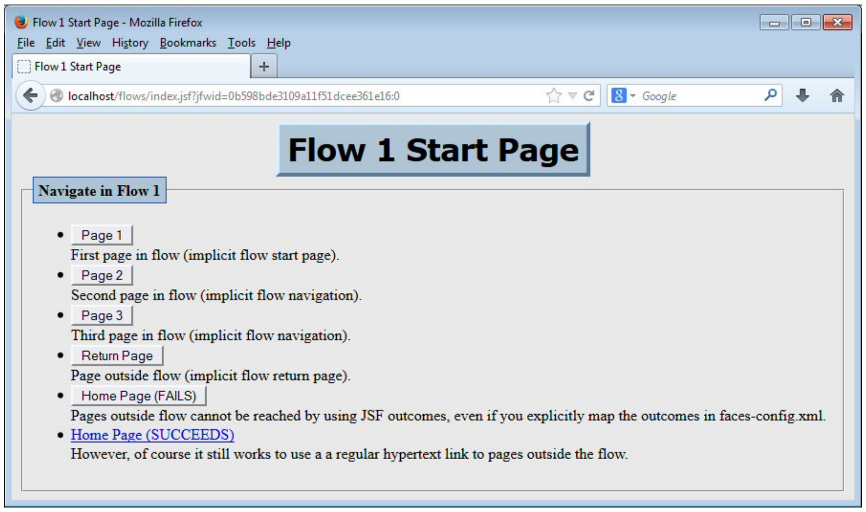
I use templates in these examples because so many of the pages have similar layouts. However, there is no particular tie between faces flow and templates.

Flow1 Template File (Excerpt)

```
...
<h:form>
<ul>
  <li><h:commandButton value="Page 1" action="flow1"/><br/>
      First page in flow (implicit flow start page).</li>
  <li><h:commandButton value="Page 2" action="page2"/><br/>
      Second page in flow (implicit flow navigation).</li>
  <li><h:commandButton value="Page 3" action="page3"/><br/>
      Third page in flow (implicit flow navigation).</li>
  <li><h:commandButton value="Return Page" action="flow1-return"/><br/>
      Page outside flow (implicit flow return page).</li>
  <li><h:commandButton value="Home Page (FAILS)" action="home"/><br/>
      Pages outside flow cannot be reached by using JSF outcomes,
      even if you explicitly map the outcomes in faces-config.xml.</li>
  <li><a href="#{request.contextPath}/index.jsf">Home Page (SUCCEEDS)</a><br/>
      However, of course it still works to use a a regular hypertext link
      to pages outside the flow.</li>
</ul>
</h:form>
...
```

Flow1 Start Page flow1/flow1.xhtml

- **Invoked by**
  - <h:commandButton … action="flow1"/>
    - From inside or outside the flow


Flow 1 Start Page - Mozilla Firefox

localhost/flows/index.jsf?jfwid=0b598bde3109a11f51dcee361e16:0

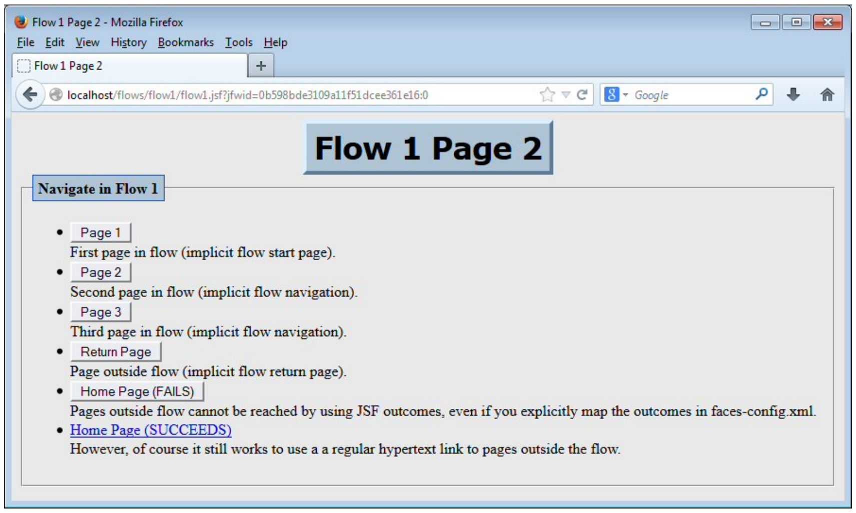
Flow 1 Start Page

Navigate in Flow 1

- Page 1
  First page in flow (implicit flow start page).
- Page 2
  Second page in flow (implicit flow navigation).
- Page 3
  Third page in flow (implicit flow navigation).
- Return Page
  Page outside flow (implicit flow return page).
- Home Page (FAILS)
  Pages outside flow cannot be reached by using JSF outcomes, even if you explicitly map the outcomes in faces-config.xml.
- Home Page (SUCCEEDS)
  However, of course it still works to use a a regular hypertext link to pages outside the flow.

Flow1 Page 2 flow1/page2.xhtml

- **Invoked by**
  - <h:commandButton … action="page2"/>
    - From inside the flow <u>only</u>


Flow 1 Page 2 - Mozilla Firefox

localhost/flows/flow1/flow1.jsf?jfwid=0b598bde3109a11f51dcee361e16:0

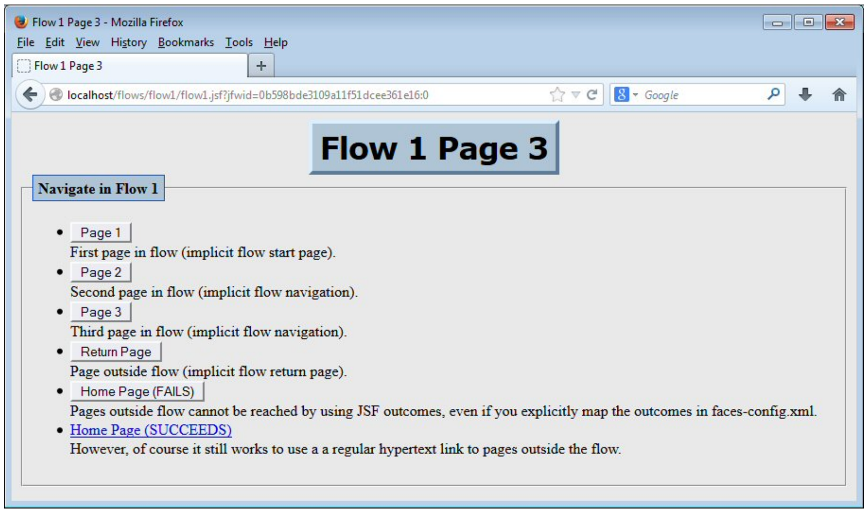
Flow 1 Page 2

Navigate in Flow 1

- Page 1
  First page in flow (implicit flow start page).
- Page 2
  Second page in flow (implicit flow navigation).
- Page 3
  Third page in flow (implicit flow navigation).
- Return Page
  Page outside flow (implicit flow return page).
- Home Page (FAILS)
  Pages outside flow cannot be reached by using JSF outcomes, even if you explicitly map the outcomes in faces-config.xml.
- Home Page (SUCCEEDS)
  However, of course it still works to use a a regular hypertext link to pages outside the flow.

Flow1 Page 3 flow1/page3.xhtml

- **Invoked by**
  - <h:commandButton … action="page3"/>
    - From inside the flow only


Flow 1 Page 3 - Mozilla Firefox

localhost/flows/flow1/flow1.jsf?jfwid=0b598bde3109a11f51dcee361e16:0

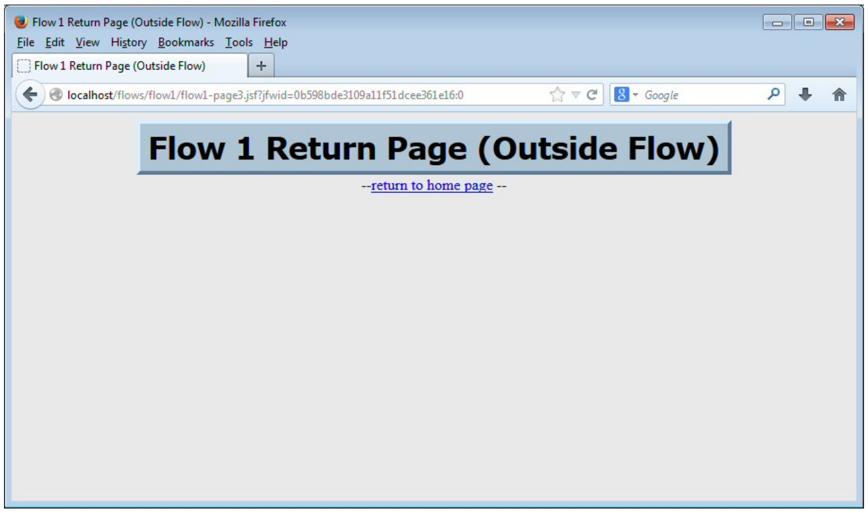
Flow 1 Page 3

Navigate in Flow 1

- Page 1
  First page in flow (implicit flow start page).
- Page 2
  Second page in flow (implicit flow navigation).
- Page 3
  Third page in flow (implicit flow navigation).
- Return Page
  Page outside flow (implicit flow return page).
- Home Page (FAILS)
  Pages outside flow cannot be reached by using JSF outcomes, even if you explicitly map the outcomes in faces-config.xml.
- Home Page (SUCCEEDS)
  However, of course it still works to use a a regular hypertext link to pages outside the flow.

Flow1 Return Page flow1-return.xhtml

- **Invoked by**
  - `<h:commandButton … action="flow1-return"/>`
    - From inside or outside the flow


Flow 1 Return Page (Outside Flow) - Mozilla Firefox

localhost/flows/flow1/flow1-page3.jsf?jfwid=0b598bde3109a11f51dcee361e16:0

Flow 1 Return Page (Outside Flow)

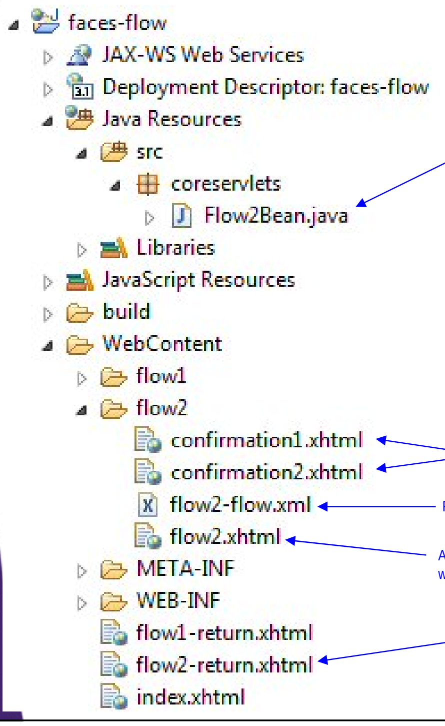
--return to home page --

Flow-Scoped Beans and the flowScope Variable

Main Points

- **Flow-scoped beans have a bigger scope than view-scoped beans**
  - They are available to the same user in all pages in a flow, rather than for a single page
- **Flow-scoped beans have a smaller scope than session-scoped beans**
  - The beans go away when user navigates out of flow
- **There is a new EL flowScope variable**
  - That can be used to directly store simple values
    - But, flow-scoped beans are usually better
- **Flow-scoped beans depend on CDI**
  - They require a Java EE 7 server or CDI to be added
  - You must use @Named, not @ManagedBean

Flow2 Example: Annotated Project Layout



```
faces-flow
  JAX-WS Web Services
  Deployment Descriptor: faces-flow
  Java Resources
    src
      coreservlets
        Flow2Bean.java
    Libraries
  JavaScript Resources
  build
  WebContent
    flow1
    flow2
      confirmation1.xhtml
      confirmation2.xhtml
      flow2-flow.xml
      flow2.xhtml
    META-INF
    WEB-INF
    flow1-return.xhtml
    flow2-return.xhtml
    index.xhtml
```

Bean is declared with @Named (not @ManagedBean) and @FlowScoped("*the-flow-id*"). It also implements Serializable, since flow-scoped beans, like view-scoped beans, are stored in the session, but are potentially removed before the session times out (e.g., flow-scoped beans are removed from the session when the user uses a JSF outcome to navigate out of the flow).

From within the flow, outcomes of "confirmation1" and "confirmation2" invoke these pages. The Flow2Bean and the flowScope variable carry values from earlier in the flow.

File must exist, but can be totally empty.

An outcome of "flow2" causes the flow to start and this page to be displayed. The Flow2Bean is instantiated when this page is visited from outside the flow.

From within the flow, an outcome of "flow2-return" invokes this page. That outcome also results in the flowScope variable being cleared and the flow-scoped Flow2Bean being made inaccessible, removed from the session, and made available for garbage collection.

Flow2Bean (Part 1)

The ID of the flow to which this bean applies. The ID is usually, but not necessarily always, the folder name.

```
...

@Named
@FlowScoped("flow2")
public class Flow2Bean implements Serializable {
  private static final long serialVersionUID = 1L;
  private String firstName, lastName;
  private int pagesViewed = 1;

  // Getters and setters for first and last name

  public int getPagesViewed() {
    return(pagesViewed++);
  }
```

Flow2Bean (Part 2)

The action controller method, as specified by <h:commandButton ... action="#{flow2Bean.doFlow}"/>
It randomly chooses a page inside the flow, assuming that the flow folder has confirmation1.xhtml and confirmation2.xhtml inside of it.

```
  public String doFlow() {
    if (Math.random() > 0.5) {
      return("confirmation1");
    } else {
      return("confirmation2");
    }
  }
}
```

Importance of Serializable

- **Flow-scoped beans are in the session**
  - So, like all things in the session, should be Serializable
    - For distributed Web apps
    - So they can live across server restarts
- **If you fail to make session-scoped beans Serializable**
  - Tomcat gives mild warning, but it works anyhow (except they do not persist across server restarts)
- **If you fail to make flow-scoped beans Serializable**
  - Tomcat fails on app startup with Weld exception

Flow2 Start Page flow2/flow2.xhtml

```
...
<h:form>
<h:panelGrid columns="3" styleClass="formTable">
  Email address:
  <h:inputText value="#{flowScope.email}" id="email"
               required="true"
               requiredMessage="Must supply email address"/>
  <h:message for="email" styleClass="error"/>

  First (given) name:
  <h:inputText value="#{flow2Bean.firstName}" id="first"
               required="true"
               requiredMessage="Must supply first name"/>
  <h:message for="first" styleClass="error"/>

  Last (family) name:
  <h:inputText value="#{flow2Bean.lastName}" id="last"
               required="true"
               requiredMessage="Must supply last name"/>
  <h:message for="last" styleClass="error"/>
</h:panelGrid>
<h:commandButton value="Submit and Go to Random Page"
                 action="#{flow2Bean.doFlow}"/>
</h:form> ...
```

You can store values directly in the flowScope variable, and this is necessary when defining inbound and outbound parameters for nested flows (see Part 2).

However, as with other scopes, beans specific to the scope are usually better.

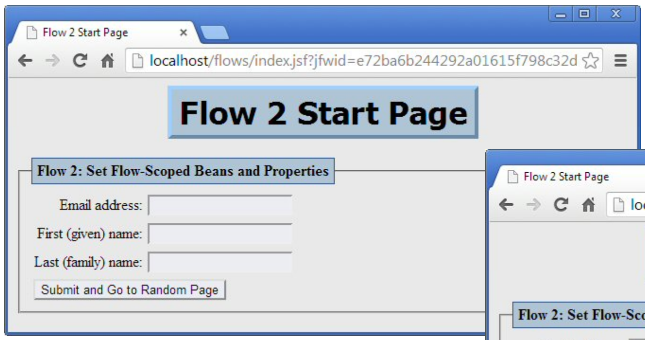
Flow2 Confirmation Pages flow2/confirmation*x*.xhtml

```
...
<h:form>
<ul>
  <li>Email address: #{flowScope.email}</li>
  <li>First name: #{flow2Bean.firstName}</li>
  <li>Last name: #{flow2Bean.lastName}</li>
  <li>Number pages viewed: #{flow2Bean.pagesViewed}</li>
  <li>Navigate:
      <h:commandLink value="Start Page" action="flow2"/>,
      <h:commandLink value="Confirmation Page 1" action="confirmation1"/>,
      <h:commandLink value="Confirmation Page 2" action="confirmation2"/>,
      <h:commandLink value="Return Page" action="flow2-return"/></li>
</ul>
</h:form>
...
```

Results: Flow2 Start Page

- **Invoked by outcome of "flow2"**
  - When arriving here from outside flow, the Flow2Bean is instantiated and placed in the session. The flowScope variable is empty at this point.

Initial state



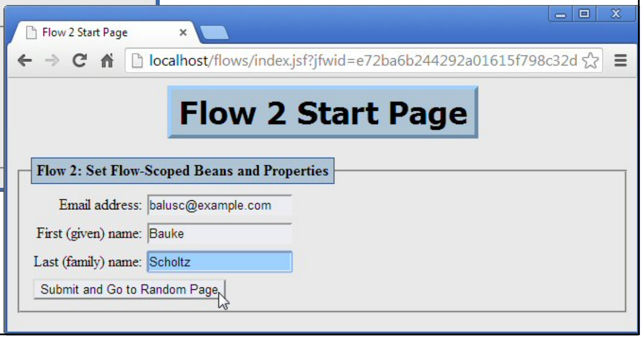
Flow 2 Start Page

Flow 2: Set Flow-Scoped Beans and Properties

Email address:
First (given) name:
Last (family) name:

Submit and Go to Random Page

After filling in



Flow 2 Start Page

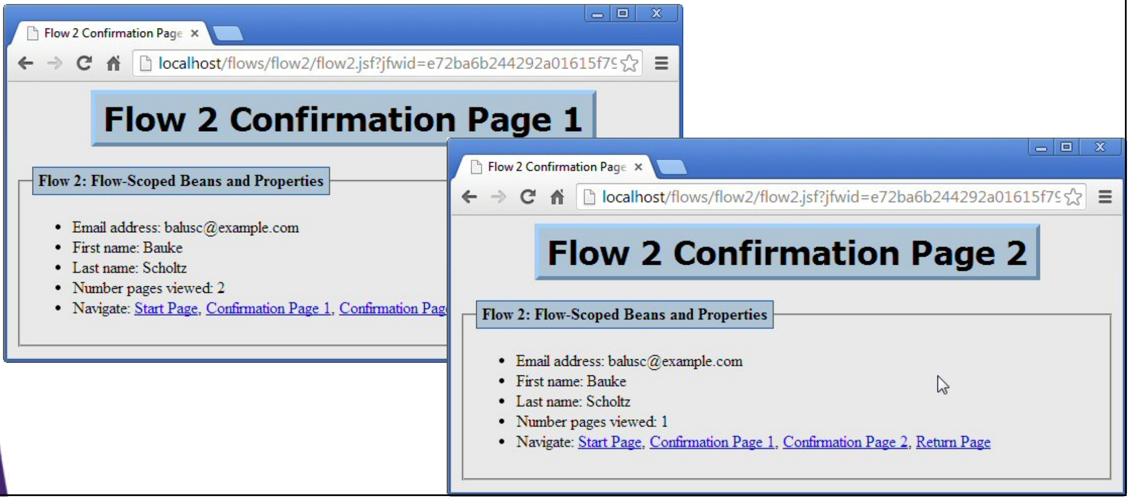
Flow 2: Set Flow-Scoped Beans and Properties

Email address: balusc@example.com
First (given) name: Bauke
Last (family) name: Scholtz

Submit and Go to Random Page

Results: Flow2 Confirmation Pages

- **Invoked by outcomes of "confirmation1" and "confirmation2"**
  – The flowScope variable and the flow2Bean carry their values from earlier pages in the flow.



Flow 2 Confirmation Page 1

Flow 2: Flow-Scoped Beans and Properties

- Email address: balusc@example.com
- First name: Bauke
- Last name: Scholtz
- Number pages viewed: 2
- Navigate: Start Page, Confirmation Page 1, Confirmation Pag

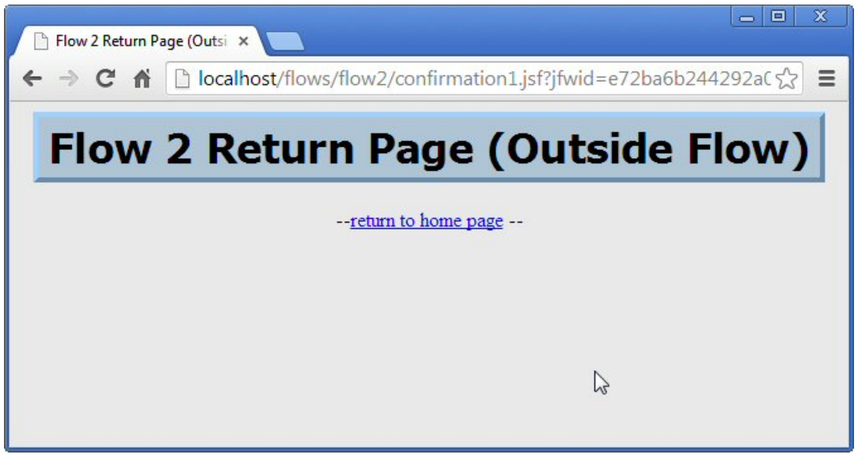
Flow 2 Confirmation Page 2

Flow 2: Flow-Scoped Beans and Properties

- Email address: balusc@example.com
- First name: Bauke
- Last name: Scholtz
- Number pages viewed: 1
- Navigate: Start Page, Confirmation Page 1, Confirmation Page 2, Return Page

Results: Flow2 Return Page

- **Invoked by outcome of "flow2-return"**
  – The flowScope variable has been cleared and the flow2Bean has been removed from the session and is no longer available at #{flow2Bean…}



Flow 2 Return Page (Outsi
localhost/flows/flow2/confirmation1.jsf?jfwid=e72ba6b244292a0

Flow 2 Return Page (Outside Flow)

--return to home page --

Configuring Flows with Flow-Specific XML Files

Main Points

- **Use *flowname/flowname*-flow.xml to**
  - Define the start page
    ```
    <start-node>startPage</start-node>
    <view id="startPage">…</view>
    ```
  - Define the return page(s) (required)
    ```
    <flow-return id="some-id">…</flow-return>
    ```
  - Map outcomes to pages
    ```
    <view id="another-id">…</view>
    ```
  - Conditionally map outcomes to pages
    ```
    <switch id="blah-id">
        <case>…</case>
        <case>…</case>
        <default-outcome>…</default-outcome>
    </switch>
    ```

XML Configuration File: General Info

- **File name and location**
  - *flowname*/*flowname*-flow.xml
- **Basic syntax**

```
<?xml version='1.0' encoding='UTF-8'?>
<faces-config … version="2.2">
    <flow-definition id="flow-id">
        …
    </flow-definition>
</faces-config>
```

Same attributes and values as regular faces-config.xml file in JSF 2.2

These entries are shown in slides in this section

- **Reminder**
  - Even if you do not use the XML file to configure the flow, the file must still exist (but can be completely empty).

Defining the Start Page (Optional)

- **Syntax**

```
<start-node>startPage</start-node>
<view id="startPage">
    <vdl-document>/flowname/start.xhtml</vdl-document>
</view>
```

- **Default**
  - If <start-node> is omitted, the usual default of /*flowname*/*flowname*.xhtml is used, as shown in previous section.

Defining the Start Page: Flow3 Example

- **Code**

```
<start-node>startPage</start-node>
<view id="startPage">
   <vdl-document>
       /flow3/flow3-starting-page.xhtml
    </vdl-document>
</view>
```

- **Notes**
  – From outside page, when outcome "flow3" occurs, the flow is started and flow3-starting-page.xhtml is invoked
  – The name “startPage” above is arbitrary. The only rule is that name inside <start-node> matches the id of some view that defines the page location.

Defining the Return Page(s) (Required)

- **Syntax**

This is the outcome. This is the page to go to when that outcome occurs.

```
<flow-return id="first-return-id">
    <from-outcome>/outside-page-1</from-outcome>
</flow-return>
<flow-return id="second-return-id">
   <from-outcome>/outside-page-2</from-outcome>
</flow-return>
```

- **No default**
  – At least one return page must be defined.
  – When using an XML configuration file, the usual default return page of *flowname*-return.xhtml goes away.

Defining the Return Pages: Flow3 Example

- **Code**

```
<flow-return id="return-from-flow3">
    <from-outcome>
        /return-page-for-flow3
    </from-outcome>
</flow-return>
<flow-return id="go-home">
    <from-outcome>/index</from-outcome>
</flow-return>
```

- **Notes**
  – It is common to have only a single return page. This example just illustrates that it is *possible* to have multiple ones.

Mapping Outcomes to Pages (Optional)

- **Syntax**

```
<view id="some-outcome">
    <vdl-document>
        /flowname/some-page.xhtml
    </vdl-document>
</view>
```

- **Default**
  – If <view> is omitted, the usual default of /*flowname*/some-outcome.xhtml is used, as in previous section.
    - Usual best practice is to either map all of the pages or use defaults for all of them, not to mix.

Mapping Outcomes to Pages: Flow3 Example

- **Code**

```
 <view id="confirmation1">
     <vdl-document>
        /flow3/confirmation-page-1.xhtml
     </vdl-document>
</view>
<view id="confirmation3">
     <vdl-document>
         /flow3/confirmation-page-3.xhtml
     </vdl-document>
</view>
```

- **Notes**
  - No mapping for "confirmation2", so default of /flow3/confirmation2.xhtml will be used.
    - Just for illustration: you usually map all pages or map none of them

Conditional Outcome Mapping: Overview

- **Idea**
  - From a single outcome, you can map to more than one page by performing conditional logic with the expression language.
  - Similar to the use of <if> within <navigation-case> in normal (non-flow) page navigation.
- **Usage is somewhat controversial**
  - The usefulness of this technique is debated. Some say to simply do the logic in the action controller method.
    - E.g., instead of returning "success" and then using <switch> to map to success-1.xhtml and success-2.xhtml, simply return "success-1" or "success-2", doing the same logic in the Java code as would have been done in the <switch> case.
  - Other say that it makes sense to do the <switch> if the logic is based on a bean *different* than the one containing the action controller method.
    - E.g., action controller returns "success", and it maps to success-newbie.xhtml and success-oldtimer.xhtml, based on a *separate* session-scoped bean that the action controller method wouldn't easily have access to.

Conditional Outcome Mapping (Optional)

- **Syntax**

```
<switch id="some-outcome">
    <case>
      <if>#{some EL test}</if>
      <from-outcome>result-id-1</from-outcome>
    </case>
    <case>
      <if>#{another EL test}</if>
      <from-outcome>result-id-2</from-outcome>
    </case>
    <default-outcome>result-id-3</default-outcome>
</switch>
```

Interpretation:

If the result was "some-outcome":
- And #{some EL test} is true, go to result-id-1 (which defaults to result-id-1.xhtml, but can be mapped with <view> as shown earlier).
- Else if #{another EL test} is true, go to result-id-2.
- Else go to result-id-3

- **Default**
  - Without <switch>, each outcome maps to a *single* page.
    - Either mapped with <view> or default of outcome.xhtml

Conditional Outcome Mapping: Flow3 Example

- **Code**

```
<switch id="confirm">
    <case>
      <if>#{flow3HitCountBean.currentCount eq 1}</if>
      <from-outcome>confirmation3</from-outcome>
    </case>
    <case>
      <if>#{flow3HitCountBean.currentCount lt 5}</if>
      <from-outcome>confirmation4</from-outcome>
    </case>
    <default-outcome>confirmation5</default-outcome>
</switch>
```

Interpretation:

If the result was "confirm":
- Newbies (first visit) go to confirmation3
- Medium-common visitors (visit 2-4) go to confirmation4
- Old-timers go to confirmation5

- **Notes**
  - The hit-count-bean is separate from the main bean.
  - Used <view> to map "confirmation3" to confirmation-page-3.xhtml
  - Used defaults for other two pages: confirmation4.xhtml, and confirmation5.xhtml.

Flow3Bean

```
package coreservlets;

import javax.faces.flow.FlowScoped;
import javax.inject.Named;

@Named
@FlowScoped("flow3")
public class Flow3Bean extends Flow2Bean {
  private static final long serialVersionUID = 1L;
}
```

Extends Flow2Bean, so same basic functionality:

- Getters/setters for first and last name
- Getter for hit count
- doFlow method that randomly returns "confirmation-1" or "confirmation-2"

But, you cannot use Flow2Bean directly, because each flow-scoped bean corresponds to a single flow.

FlowHitCount3Bean

```
...
@Named
@SessionScoped
public class Flow3HitCountBean implements Serializable {
  private static final long serialVersionUID = 1L;
  private int hitCount;

  public int getCurrentCount() {
    return(hitCount);
  }

  public int getIncrementingCount() {
    return(++hitCount);
  }
}
```

This is the bean that is used in the “switch” shown earlier. It assumes that the flow3 start page (and no other) calls #{flow3HitCountBean.incrementingCount}.

Note also that If you use @Named instead of @ManagedBean, then you must use the CDI (javax.enterprise.context) version of SessionScoped, not the JSF version (javax.faces.bean). With faces flow, you use CDI consistently.

Flow3 Configuration File (Part 1) flow3/flow3-flow.xml

```
<?xml version='1.0' encoding='UTF-8'?>
<faces-config
    xmlns="http://xmlns.jcp.org/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
         http://xmlns.jcp.org/xml/ns/javaee/web-facesconfig_2_2.xsd"
    version="2.2">

  <flow-definition id="flow3">
    <start-node>startPage</start-node>
    <view id="startPage">
      <vdl-document>/flow3/flow3-starting-page.xhtml</vdl-document>
    </view>
    <flow-return id="return-from-flow3">
      <from-outcome>/return-page-for-flow3</from-outcome>
    </flow-return>
    <flow-return id="go-home">
      <from-outcome>/index</from-outcome>
    </flow-return>
```

Flow3 Configuration File (Part 2) flow3/flow3-flow.xml

```
    <view id="confirmation1">
      <vdl-document>/flow3/confirmation-page-1.xhtml</vdl-document>
    </view>
    <view id="confirmation3">
      <vdl-document>/flow3/confirmation-page-3.xhtml</vdl-document>
    </view>

    <switch id="confirm">
      <case>
        <if>#{flow3HitCountBean.currentCount eq 1}</if>
        <from-outcome>confirmation3</from-outcome>
      </case>
      <case>
        <if>#{flow3HitCountBean.currentCount lt 5}</if>
        <from-outcome>confirmation4</from-outcome>
      </case>
      <default-outcome>confirmation5</default-outcome>
    </switch>

 </flow-definition>
</faces-config>
```

Flow3 Start Page flow3/flow3-starting-page.xhtml

```
...<h:form>
<h:panelGrid columns="3" styleClass="formTable">
  Email address:
  <h:inputText value="#{flowScope.email}" id="email"
               required="true"
               requiredMessage="Must supply email address"/>
  <h:message for="email" styleClass="error"/>

  First (given) name:
  <h:inputText value="#{flow3Bean.firstName}" id="first"
               required="true"
               requiredMessage="Must supply first name"/>
  <h:message for="first" styleClass="error"/>

  (Similar entry for last name)

  <f:facet name="footer">
    <h:commandButton value="Random Confirmation Page (via Java)"
                     action="#{flow3Bean.doFlow}"/><br/>
    <h:commandButton value="Confirmation Page Based on Visits (via switch)"
                     action="confirm"/>
  </f:facet>
</h:panelGrid>
</h:form>...
```

Returns either "confirmation1" or "confirmation2" at random; "confirmation1" is mapped via the <view> tag to confirmation-page-1.xhtml, and "confirmation2" has the default mapping of confirmation2.xhtml.

The "confirm" entry corresponds to the <switch> tag, and, depending on the number of visits, is mapped to "confirmation3", "confirmation4", or "confirmation5". "confirmation3" is mapped via the <view> tag to confirmation-page-3.xhtml, whereas "confirmation4" and "confirmation5" have the default mappings of confirmation4.xhtml and confirmation5.xhtml.

Flow3 Confirmation Pages

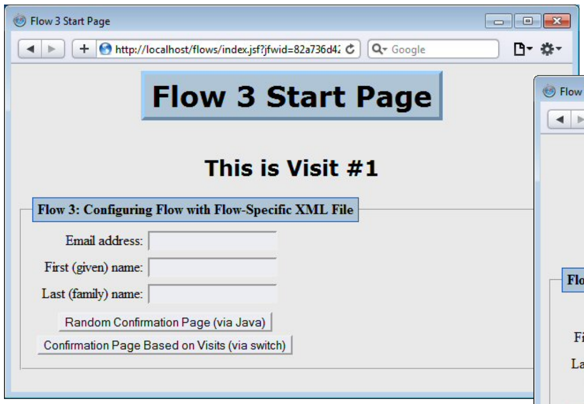
```
...
<h:form>
<ul>
  <li>Email address: #{flowScope.email}</li>
  <li>First name: #{flow3Bean.firstName}</li>
  <li>Last name: #{flow3Bean.lastName}</li>
  <li>Number pages viewed: #{flow3Bean.pagesViewed}</li>
  <li>Navigate:
      <h:commandLink value="Start Page" action="startPage"/>,
      <h:commandLink value="Confirmation Page 1" action="confirmation1"/>,
      <h:commandLink value="Confirmation Page 2" action="confirmation2"/>,
      <h:commandLink value="Confirmation Page 3" action="confirmation3"/>,
      <h:commandLink value="Confirmation Page 4" action="confirmation4"/>,
      <h:commandLink value="Confirmation Page 5" action="confirmation5"/>,
      <h:commandLink value="Return Page" action="return-from-flow3"/>,
      <h:commandLink value="Home Page" action="go-home"/></li>
</ul>
</h:form>
...
```

The five confirmation pages differ only in their titles.

Results: Flow3 Start Page

- **Invoked by outcome of "flow3"**
  - When arriving here from outside flow, the Flow3Bean is instantiated and placed in the session. The flowScope variable is empty at this point.

Initial state



Flow 3 Start Page

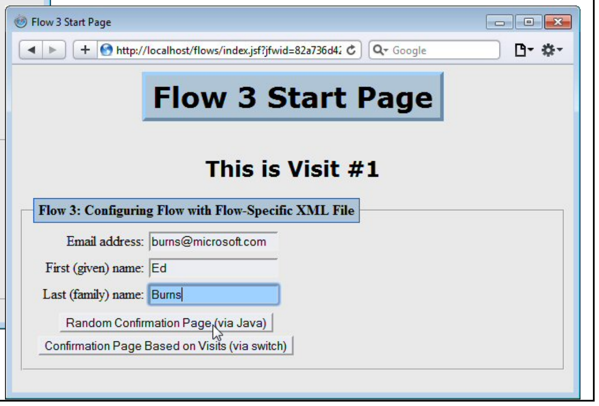
This is Visit #1

Flow 3: Configuring Flow with Flow-Specific XML File

Email address:	
First (given) name:	
Last (family) name:	

Random Confirmation Page (via Java)
Confirmation Page Based on Visits (via switch)

After filling in



Flow 3 Start Page

This is Visit #1

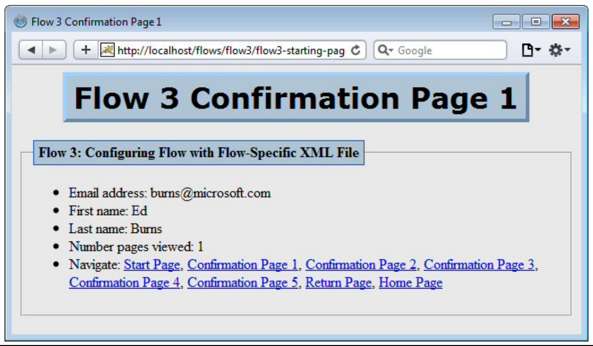
Flow 3: Configuring Flow with Flow-Specific XML File

Email address:	burns@microsoft.com
First (given) name:	Ed
Last (family) name:	Burns

Random Confirmation Page (via Java)
Confirmation Page Based on Visits (via switch)

Results: Flow3 Confirmation Pages

- **Invoked by outcomes of "confirmation1", "confirmation2", and "confirm"**
  - Pressing the top button on start page runs the doFlow method, which randomly chooses "confirmation1" or "confirmation2". Pressing the bottom button sends "confirm", which is mapped by the switch to "confirmation3", "confirmation4", and "confirmation5".


Flow 3 Confirmation Page 1
http://localhost/flows/flow3/flow3-starting-pag
Google

Flow 3 Confirmation Page 1

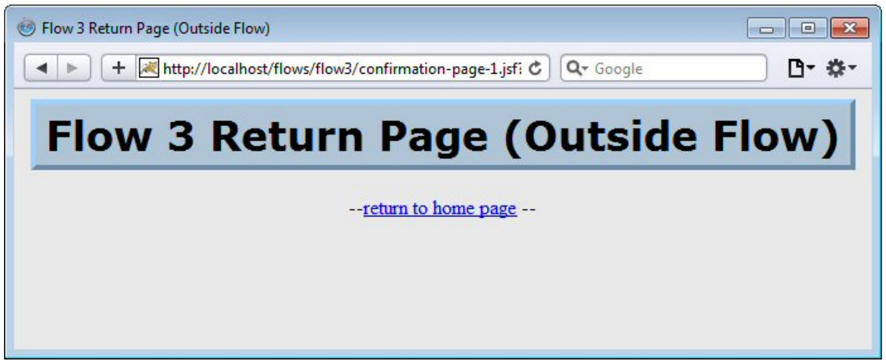
Flow 3: Configuring Flow with Flow-Specific XML File

- Email address: burns@microsoft.com
- First name: Ed
- Last name: Burns
- Number pages viewed: 1
- Navigate: Start Page, Confirmation Page 1, Confirmation Page 2, Confirmation Page 3, Confirmation Page 4, Confirmation Page 5, Return Page, Home Page

The other four confirmation pages are similar

Results: Flow3 Return Pages

- **Invoked by "return-from-flow3" or "go-home"**
  - Two return pages are defined in flow3-flow.xml, as shown earlier.
    - The first results in the return page shown below. The second results in the home page for the flows app.


Flow 3 Return Page (Outside Flow)
http://localhost/flows/flow3/confirmation-page-1.jsf
Google

Flow 3 Return Page (Outside Flow)

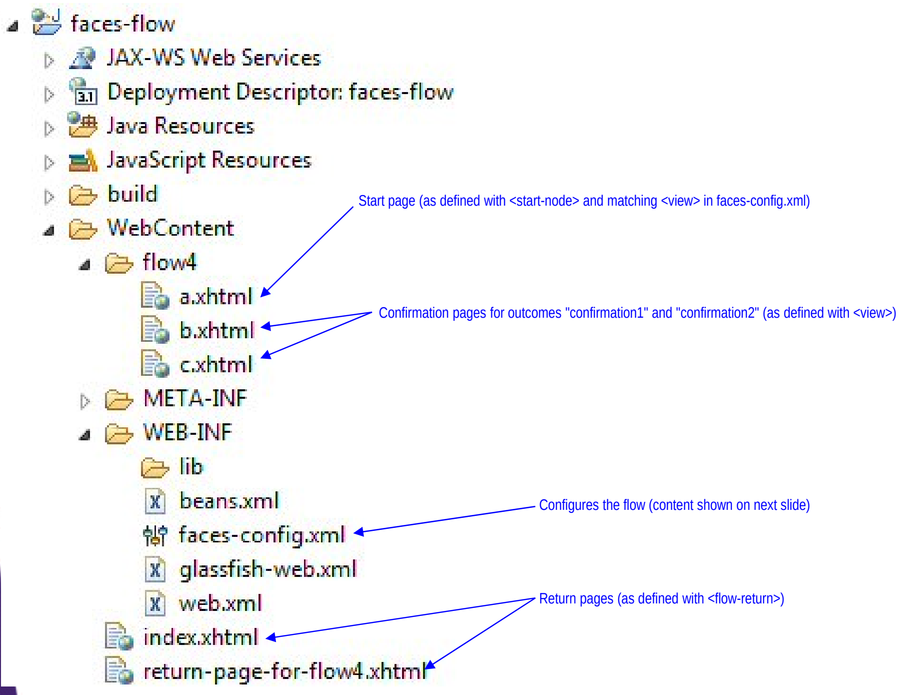
--return to home page --

Configuring Flows with the Global XML File

Main Points

- **Anything you can do with *flowname*-flow.xml, you can do with faces-config.xml**
  - Define a custom start page
    - With <start-node> and matching <view>
  - Define return pages
    - With <flow-return>
  - Map outcomes to pages
    - With <view> and <vdl-document>
  - Do conditional outcome mapping
    - With <switch>
  - Define nested flows, outbound parameters, and inbound parameters (discussed in Part 2 of faces flow tutorial)
    - With <flow-call>, <outbound-parameter>, and <inbound-parameter>

Flow4 Example: Annotated Project Layout



```
faces-flow
  JAX-WS Web Services
  Deployment Descriptor: faces-flow
  Java Resources
  JavaScript Resources
  build
  WebContent
    flow4
      a.xhtml
      b.xhtml
      c.xhtml
    META-INF
    WEB-INF
      lib
      beans.xml
      faces-config.xml
      glassfish-web.xml
      web.xml
    index.xhtml
    return-page-for-flow4.xhtml
```

Start page (as defined with <start-node> and matching <view> in faces-config.xml)

Confirmation pages for outcomes "confirmation1" and "confirmation2" (as defined with <view>)

Configures the flow (content shown on next slide)

Return pages (as defined with <flow-return>)

Flow4 Flow Definition WEB-INF/faces-config.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<faces-config ... version="2.2">...
  <flow-definition id="fourth-flow">
    <start-node>startPage</start-node>
    <view id="startPage">
      <vdl-document>/flow4/a.xhtml</vdl-document>
    </view>
    <view id="confirmation1">
      <vdl-document>/flow4/b.xhtml</vdl-document>
    </view>
    <view id="confirmation2">
      <vdl-document>/flow4/c.xhtml</vdl-document>
    </view>
    <flow-return id="return-from-flow4">
      <from-outcome>/return-page-for-flow4</from-outcome>
    </flow-return>
    <flow-return id="go-home">
      <from-outcome>/index</from-outcome>
    </flow-return>
  </flow-definition>
</faces-config>
```

Wrap-Up

Summary

- **Conventions**
  - Folder must contain *flowname*-flow.xml (can be empty)
  - Start page is *flowname*/*flowname*.xhtml
  - Outcomes within flow map to *flowname*/*outcome*.xhtml
  - Return page is *flowname*-return.xhtml
- **Flow-scoped beans**
  - Use @Named and @FlowScoped("*flowname*")
- **XML configuration file**
  - Custom start page: <start-node> and matching <view>
  - Return pages: <flow-return>
  - Mapping outcomes to pages: <view> & <vdl-document>
  - Conditional outcome mapping: <switch>
  - Usually use *flowname*-flow.xml, but can use faces-config

Faces Flow Part 2 (Next Section)

- **Nested flows**
  - Calling a subflow: uses <flow-call>
  - Sending values to subflow: uses <outbound-parameter>
  - Receiving values in subflow: uses <inbound-parameter>
  - Returning from subflow: exits to a page of calling flow
- **Defining flows with Java**
  - Uses many annotations to set things up. Uses a FlowBuilder.
  - Defines pages with builder.viewNode(…)
  - Defines start with builder.viewNode(…). markAsStartNode()
  - Defines return pages with builder.returnNode(…)
  - Defines switches with builder.switchNode(…)
  - Can define nested flows and in/outbound parameters

Questions?

More info:
http://www.coreservlets.com/JSF-Tutorial/jsf2/ – JSF 2.2 tutorial
http://www.coreservlets.com/JSF-Tutorial/primefaces/ – PrimeFaces tutorial
http://courses.coreservlets.com/jsf-training.html – Customized JSF and PrimeFaces training courses
http://coreservlets.com/ – JSF 2, PrimeFaces, Java 7 or 8, Ajax, jQuery, Hadoop, RESTful Web Services, Android, HTML5, Spring, Hibernate, Servlets, JSP, GWT, and other Java EE training