Spring Web Flow

Mastering Workflows in the Browser with Spring Web Flow

Topics

- What is Web Flow and when should I use it or not?
- Configuring Web Flow and integrating it into a Spring web application.
- Application context with JSP and Tiles.
- Real world business problems and their web flow solutions.
  - Purchasing Insurance Online.
- Planning a flow - flow variables, flow scopes.
- A new kind of IOC, Inversion of the Controller. The flow.xml.
- Service instance and a great big flow state object.
- Passing control to a subflow and back.
- Staying on the page with two kinds of AJAX.
- Advanced topics will include:
  - A Web Flow "Transaction"
  - shutting off the back button
  - listening to a flow
  - Getting flow information from outside the flow.

What is Web Flow and when should I use it or not?

Enforced/Controlled navigation.

Workflow for the web.

I think wizard.

Configuring Web Flow and integrating it into a Spring web.

- Create a new app with roo.
- Use my empty project.
- Add web-config.xml to an existing.
- Tiles are nice.

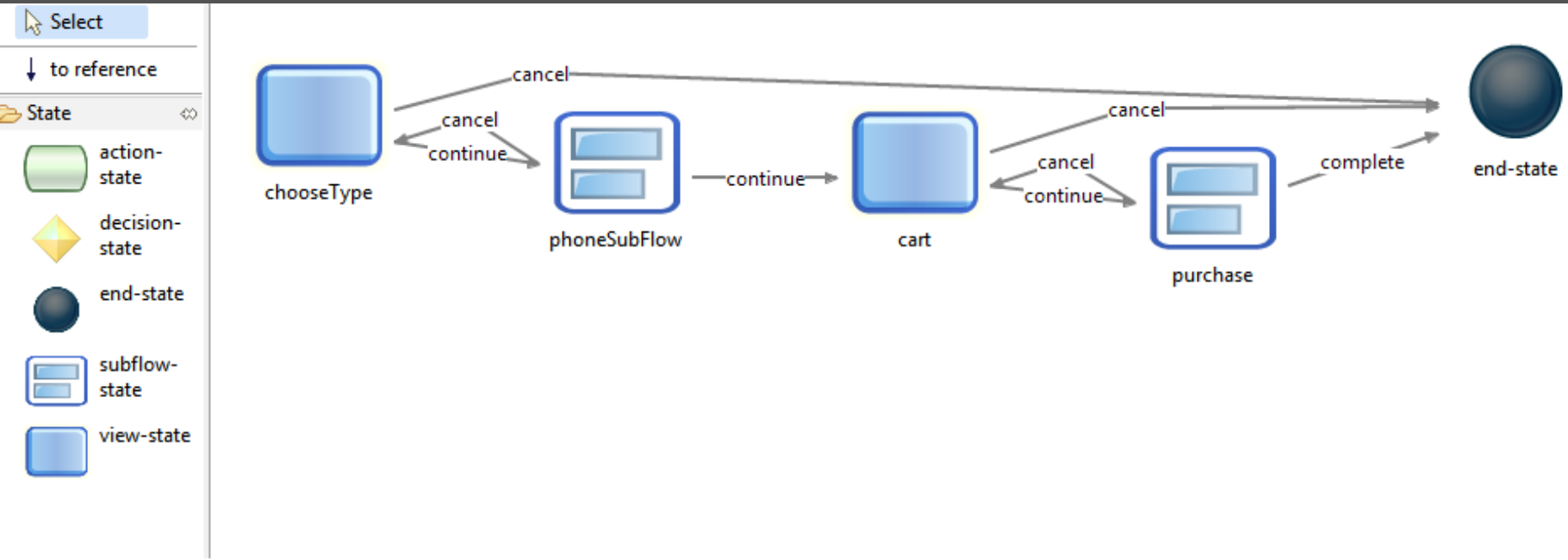
Documentation

API

The Real World

Blue KC

Planning a flow - Visualization



Inversion of the controller

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<flow xmlns="http://www.springframework.org/schema/webflow" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.springframework.org/schema/webflow http://www.springframework.org/schema/webflow/spring-webflow-2.0.xsd">

	<view-state id="cordless" view="cordlessphone/main">
		<transition on="cancel" to="cancel"/>
	</view-state>

	<end-state id="cancel" />

</flow>
```

Tags

- view-state
- transition
- global-transition
- render fragments=
- end-data
- subflow-state
- history = invlidate, discard
- action-state
- decision-state
- on-start
- on-entry
- on-exit
- on-end

Scope and Control

- request
- flash
- request
- view
- flow
- conversation
- session from externalContext

Special Variables: flowScope, viewScope, requestScope, flashScope, conversationScope, requestParameters, currentEvent, currentUser, messageContext, resourceBundle, flowRequestContext, flowExecutionContext, flowExecutionUrl, externalContext.

Subflows, Big ol state object, validation etc.

Build Your Own

Calling the flow event

```
<a href="${flowExecutionUrl}&amp;_eventId=next">Or click here
for a standard phone</a>
```

form:form - Getting data from the page.

```
<form:form method="POST" commandName="choose">
        <div class="submit">
                <label>Choose a type of phone</label>
                <form:select path="type">
                        <form:option value="stdphone">Standard Phone</form:option>
                        <form:option value="cordlessphone">Cordless Phone</form:option>
                </form:select>
                <input type="submit" id="cancel" name="_eventId_cancel" value="cancel" />
                <input type="submit" id="success" name="_eventId_continue" value="continue" />
        </div>
</form:form>
…
<view-state id="chooseType" view="mainflow/choose-type" model="choose">
        <on-entry>
                <evaluate expression="new org.demo.buildyourown.web.webflow.command.Choose()" result="viewScope.choose"/>
        </on-entry>
        <transition on="continue" to="phoneSubFlow">
                <evaluate expression="mainService.makePhone(choose)" result="conversationScope.phone" />
                <set name="requestScope.nextFlowName" value="mainService.nextFlowName(phone)" />
        </transition>
        <transition on="cancel" to="end-state"/>
</view-state>
…

public class Choose implements Serializable{
  public String getType() {
        return type;
  }
  public void setType(String type) {
        this.type = type;
  }
}
```

Validating the form data.

In a bean use: `public void validateChooseColor(ValidationContext context)` where ChooseColor is the state id.

or use JSR 303 annotations.

or create a validation class:

```
@Component
public class ColorChooserValidator
```

where colorChooser is the model object name.
Add methods for each view-state you want to validate…

```
public void validateChooseColor(ColorChooser color, ValidationContext context)
```

where ChooseColor is the State id.

Ajax...

```
<script type="text/javascript">
        Spring.addDecoration(new Spring.AjaxEventDecoration({
                elementId: "styleId1",
                formId: "styleForm",
                event: "onclick",
                params: {_eventId : "changeStyle", fragments : "body"}
        }));
</script>


...

<view-state id="chooseStyle" view="stdphone/style" model="styleChooser">
        <on-entry>
                <evaluate expression="new org.demo.buildyourown.web.webflow.command.StyleChooser()" result="viewScope.styleChooser" />
        </on-entry>
        <transition on="changeStyle" validate="false">
                <render fragments="body" />
        </transition>
        <transition on="continue" to="continue">
                <evaluate expression="stdPhoneService.addStyle(phone, styleChooser)" />
        </transition>
        <transition on="cancel" to="cancel"/>
</view-state>
```

Own your response: ajax, pdf, doc...

```
<transition on="editIt" bind="false" validate="false">
                <!-- This method never returns to a view.. -->
                <evaluate expression="editItService.editIt(flowRequestContext, currentTransaction)" />

</transition>

…
function(model) {
        var $promise = $.Deferred();
        $
                .ajax({
                        type : "POST",
                        processData : false,
                        contentType : "application/json",
                        url : flowExecutionUrl + "&_eventId=editIt&ajaxSource=true",
                        data : JSON.stringify(model),
                        dataType : "json",
                        success : function(content) {
                                $promise.resolve(content);
                        },
                });
        return $promise;};
…

public void editIt(RequestContext context, ItTransaction currentTransaction){
        ServletResponse response = (ServletResponse) context.getExternalContext().getNativeResponse();
        //write to the response...
        externalContext.recordResponseComplete();
}
```

Advanced topics...

- A Web Flow “Transaction” is shutting off the back button, but you can’t shut off the back button.
- Flow Listeners.
- Super fancy when you have to be.

end-state

Questions?