Directing Users Based on Role with Spring Web Flow

INTRODUCTION

Sekhar Srinivasan

@sekharonline4u www.sekhartheguru.net

Overview

What is Spring Web Flow

Elements of Spring Web Flow

Configure Web Flow

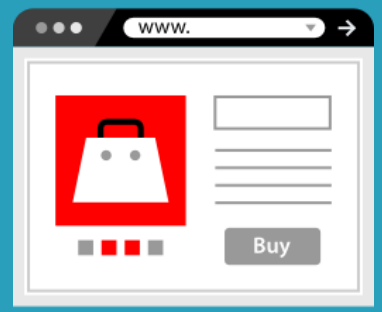
Directing Users Based on Roles

Understanding Spring Web Flow

Understanding Spring Web Flow

1. Why we need Spring Web Flow?
2. What is Spring Web Flow?
3. Advantages of Spring Web Flow

Why we need Spring Web Flow?



3 Core Problems

How do we express Page Navigation Rules?

How do we manage Navigation and Conversational State?

How do you facilitate modularization and reuse?

Spring Web Flow

What is Spring Web Flow?

Spring Web Flow is Sub-Project of Spring Framework which focuses to model and manage the Web Applications UI flow.

Separates the definition of an application's flow from the classes and views.

Captures the navigation rules by allowing the Spring Web Flow Execution engine to manage a conversation and associated state.

Spring Web Flow is reusable Web Application module

Advantages of Spring Web Flow

- Page flow of application is clearly visible
- Spring Web Flows are self-contained
- A technique to capture the page flow in a web application
- Clear, observable lifecycle and managed automatically
- System manages the complexity
- Providing infrastructure for building rich web applications are easy
- Web flow is a well-defined contract for use

Configuring Spring Web Flows

Configuring Spring Web Flows

Download and add required JAR files

Configure Spring Web Flow

Configure Flow Executor

Configure Flow Registry

Handle Flow Requests

1

Download Jar Files

Spring-Binding

Spring-JS

Spring-Webflow

Configure Spring Web Flow in Spring Application Context

Configure Flow Executor

Flow Executor is responsible to drive the execution of a flow.

Configure Flow Registry

Flow Registry will load the flow definitions to make them available to Flow Executor

```
<flow:flow-registry id="flowRegistry" base-path="/WEB-
INF/flows">


    <flow:flow-location-pattern value="**/*-flow.xml" />


</flow:flow-registry>
```

Configure Flow Registry

If Spring Web Flows are defined in Multiple Files

```
<flow:flow-registry id="flowRegistry"
                    base-path="/WEB-
                    INF/flows" >

    <flow:flow-location-pattern value="**/*-flow.xml" />
    INF/flows/bank-flow.xml />

</flow:flow-registry>
```

Configure Flow Registry

For Single Web Flow

Handle Flow Requests

Spring Web Flow provides Spring MVC handler adapter called as FlowHandlerAdapter

Handle Flow Requests

Spring Web Flow provides Spring MVC handler adapter called as FlowHandlerAdapter

HandlerAdapter is the bridge between DispatcherServlet and the Spring Web Flow

Handle Flow Requests

Spring Web Flow provides Spring MVC handler adapter called as FlowHandlerAdapter

HandlerAdapter is the bridge between DispatcherServlet and the Spring Web Flow

Handles the flow requests and manipulates the flow based on those requests.

Elements of Spring Web Flow

Elements of Spring Web Flow

States

Transitions

Flow Data

States

Application usually performs some logic based on the user input and then a decision is made to determine the next step to take and the points in the flow where these things happens are known as States.

Spring Web Flow defines 5 different kinds of State

1

```
<view-state id="login" view="Views/login.jsp"
model="loginBean">

    <transition on="performLogin" to="doLoginAction" />

</view-state>
```

<view-state />

Used to display information to a user and allow the user to play an active role in the flow by obtaining the user input using a form.

The actual view implementation could be any of the views that are supported by spring MVC

2

```
<action-state id="doLoginAction">
    <evaluate
expression="loginService.validateUser(loginBean)" />
    <transition on="true" to="showAccounts" />
    <transition on="false" to="error" />
</action-state>
```

<action-state>

State where the logic of a flow takes place and used to control the execution of an action at a point within the flow.

<evaluate> describes what needs to be done

3

```
<decision-state id="isAccountExists">
    <if test="accountService.isAccountExists()"
        then="warning"
        else="createAccount" />
</decision-state>
```

<decision-state>

State where decision is made.

A decision-state has two transitions or directions

Routing of the flow will be based on the decision evaluation of flow data.

4

```
<subflow-state id="newAccount" subflow"duplicateAccount">
  <transition on="accountNotFound" to="createAccount">
    <evaluate
expression="accountService.createAccount(accountBean)"/>
  </transition>
  <transition on="duplicateAccount" to="warning" />
</subflow-state>
```

<subflow-state>

Starts a new flow within the context of a flow that is already running, and the sub flow returns to the original flow when it is completed.

Data may be passed from the calling flow into the sub flow and also the output data from the sub flow may be retrieved into the calling flow.

5

```
<end-state id="accountCreated" />
```

<end-state>

Used to specify end of the flow when entered.

If <end-state> is a part of sub flow then root flow is resumed.

Transitions and Flow Data Elements of Spring Web Flow

What are Transitions?

Transitions are used to
connect the states in a Flow

Transitions are used to connect the states in a Flow

Transition elements are used to handle the events that occur within a state

Transitions are used to connect the states in a Flow

Transitions elements are used to handle the events that occur within a state

Transition element is a child-element of various state elements such as view-state, action-state, decision-state

Transitions are used to connect the states in a Flow

Transitions elements are used to handle the events that occur within a state

Transition element is a child-element of various state elements such as view-state, action-state, decision-state

Every state in a flow other than the end-state should have at-least one transition

Transitions are used to connect the states in a Flow

Transitions elements are used to handle the events that occur within a state

Transition element is a child-element of various state elements such as view-state, action-state, decision-state

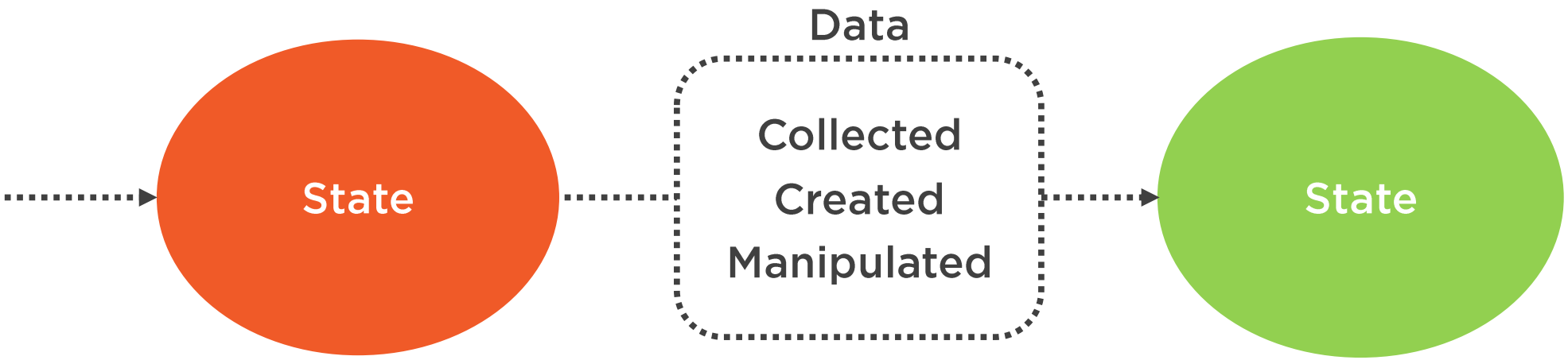
Every state in a flow other than the end-state should have at-least one transition

State may have any no. of transitions where each transition is used to represent a different path

```
<action-state id="deleteAccount">
    <evaluate expression="accountService.deleteAccount(
                          requestParameters.accountNo); />
    <transition on="true" to="listAccounts" />
</action-state>
```

To Define a Transition

Understanding Flow Data



Flow Data is stored in variables that can be referenced at various points in the flow

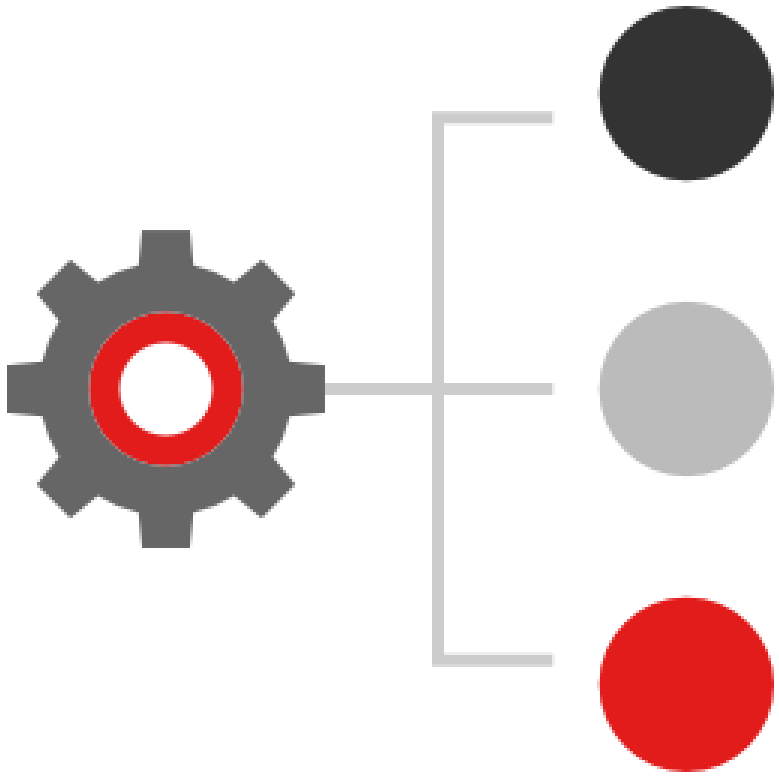
```
<var name="accountBean"
class="com.ps.psbankapp.model.AccountBean" />


<set name="flowScope.account"
        value="new com.ps.psbankapp.model.AccountBean()" />
```

Creating Flow Data

Spring Web Flow supports 5 different scopes for the Flow Data
Spring Web Flow variables have varying lifespans, visibility depending on the scope it belongs to

Scopes of Flow Data



Flow Scope

- Created at the beginning of a flow and is destroyed when flow ends

Conversation Scope

- Created at the beginning of a top-level flow and is destroyed when the flow ends

Request Scope

- Created at the beginning of an Http Request and destroyed at the end of request

Flash Scope

- Created when a flow begins, wiped clean when a view is rendered and destroyed at the end of a flow

View Scope

- Created when the flow enters a view-state and destroyed when the view is rendered

Elements of Web Flow

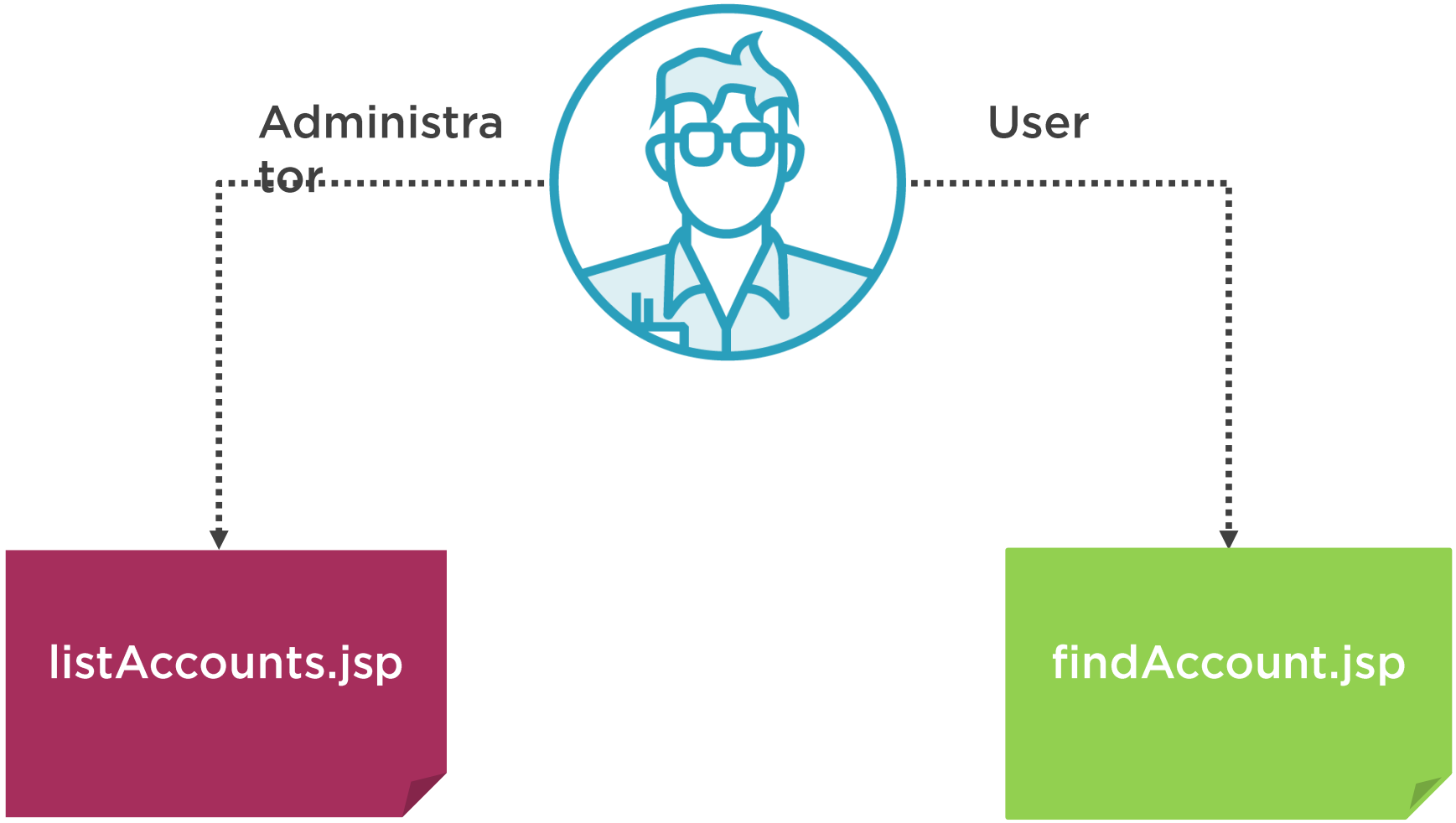
States

Transitions

Flow Data

Demo: Creating a Web Flow

Motivation



Demo: Adding Authentication Success Handler

Summary

- Introduction to Spring Framework
- How to Configure our environment for Spring MVC Application
- Create Controllers and Views
- Handle Spring Tags and Data Bindings
- Handle Request Parameters and Request Mappings
- Applying Built-in Validation Rules
- Perform CRUD Operations using Hibernate
- Manage Exceptions using AOP
- Modified Front-end using Bootstrap
- Creating REST Services
- Redirecting the Users based on the Roles with Spring Web Flow