Spring MVC 2.5 and Beyond

Covering Core Spring MVC and official Spring MVC Extensions

Rossen Stoyanchev

Senior Consultant, SpringSource

Agenda

- Core Spring MVC 2.5
  - New @Controller model
  - More convention-over-configuration
- The New Spring Web Flow 2.0
  - Simplified flow definition language
  - Ajax event model
  - Full JSF support

Spring MVC

- Spring MVC is a popular web framework
- Spring MVC is the platform for developing Spring-based web applications
  - All extensions like Web Flow plug into it
- Spring MVC 2.5 greatly simplifies the Controller programming model through use of annotations and convention-over-configuration
  - "Spring @MVC" has emerged as a catch phrase for talking about these new features

@Controller - What is it?

- A new POJO-based "multi-action" controller model
  - With support for processing forms
  - Can fully supercede use of the SimpleFormController and MultiActionController classes

@Controller - How Does It Work?

- Define your Controllers as simple POJO classes instead of extending from framework-specific base classes
- Map URLs to action methods using annotations
- Declare action method arguments to bind HTTP request data

Example @Controller

```
@Controller
public class HotelsController {

    @RequestMapping
    public void index() {…}

    @RequestMapping
    public void search(SearchCriteria criteria,
                       BindingResult result) {…}

    @RequestMapping
    public void show() {…}
}
```

Equivalent Spring 2.0 Example

```
public class HotelsController extends
   MultiActionController {

   public ModelAndView index(HttpServletRequest req,
    HttpServletResponse res) {…}

   public ModelAndView search(HttpServletRequest req,
    HttpServletResponse res) {…}

   public ModelAndView show(HttpServletRequest req,
    HttpServletResponse res) {…}

}
```

URL->@Controller Method Mapping

- Mapping is based on the request path
- Can also use the request method
- The strategies for mapping are
  - Simple
  - Controller Relative
  - Externalized Controller Relative
  - Convention-based

Simple

```
@Controller
public class HotelsController {
    @RequestMapping("/hotels/index")
    public void index() {…}

    @RequestMapping("/hotels/search")
    public void search() {…}

    @RequestMapping("/hotels/show")
    public void show() {…}
}
```

Controller Relative

```
@Controller
@RequestMapping("/hotels/*")
public class HotelsController {
    @RequestMapping(method=RequestMethod.GET)
    public void index() {…}

    @RequestMapping
    public void search() {…}

    @RequestMapping(method=RequestMethod.GET)
    public void show() {…}
}
```

Externalized Controller Relative

```
<bean class="...SimpleUrlHandlerMapping">
    <property name="mappings">
        <value>
            /hotels/*=hotelsController
        </value>
    </property>
</bean>
```

```
@Controller
public class HotelsController {
    @RequestMapping(method=RequestMethod.GET)
    public void index() {…}

    @RequestMapping
    public void search() {…}

    @RequestMapping(method=RequestMethod.GET)
    public void show() {…}
}
```

Convention Based

```
<bean class="...ControllerClassNameUrlHandlerMapping" />

@Controller
public class HotelsController {
    @RequestMapping(method=RequestMethod.GET)
    public void index() {…}

    @RequestMapping
    public void search() {…}

    @RequestMapping(method=RequestMethod.GET)
    public void show() {…}
}
```

Action Method Binding

- You can bind HTTP Request data to action method parameters, including
  - Simple parameter types
  - Bean parameter types
  - Several special parameter types

Simple Parameter Types

```
@RequestMapping
public void show(@RequestParam("id") Long id) {…}
```

Bean Parameter Types

```
@RequestMapping
public void search(SearchCriteria criteria) {…}
```

Special Parameter Types

```
@RequestMapping
public void foo(HttpServletRequest req, …) {…}

@RequestMapping
public void foo(HttpServletResponse res, …) {…}

@RequestMapping
public void foo(HttpSession session, …) {…}

@RequestMapping
public void foo(Locale locale, …) {…}
```

Special Parameter Types (2)

```
@RequestMapping
public void foo(Model model, …) {…}

@RequestMapping
public void search(SearchCriteria criteria,
                   BindingResult bindingResult, …) {…}

@RequestMapping
public void foo(SessionStatus status, …) {…}
```

Selecting Views

- Your controllers can select views to render
  - By convention (the default)
  - By returning a view name string

Selecting a View by Convention

```
@Controller
public class HotelsController {
    @RequestMapping
    public void index() {…}
}
```

- The request path is used as the view name by default
  - Assume a request URL of `/hotels/index`
  - The view rendered will be `/WEB-INF/hotels/index.jsp`
  - Customize this convention with a RequestToViewNameTranslator
  - Custom default view resolution rules with a ViewResolver

Selecting a View Explicitly

```
@Controller
public class HotelsController {
    @RequestMapping
    public String index() { … return "/hotels/index"}
}
```

- The name of the view to render is returned
- Customize how the view is resolved with a ViewResolver

Exporting @Controllers

- How do get your @Controller exported to the web?
- Two techniques
  - Explicit bean definition
  - Classpath Scanning

Explicit Bean Definition

```
<bean class="...SimpleUrlHandlerMapping">
    <property name="mappings">
        <value>
            /hotels/*=hotelsController
        </value>
    </property>
</bean>

<bean id="hotelsController" class="…HotelsController" />
```

Classpath Scanning

```
<!-- Activates annotation-based configuration -->
<context:annotation-config />

<!-- Scans for @Components to export and configure -->
<context:component-scan base-package="com.mycompany.app"/>
```

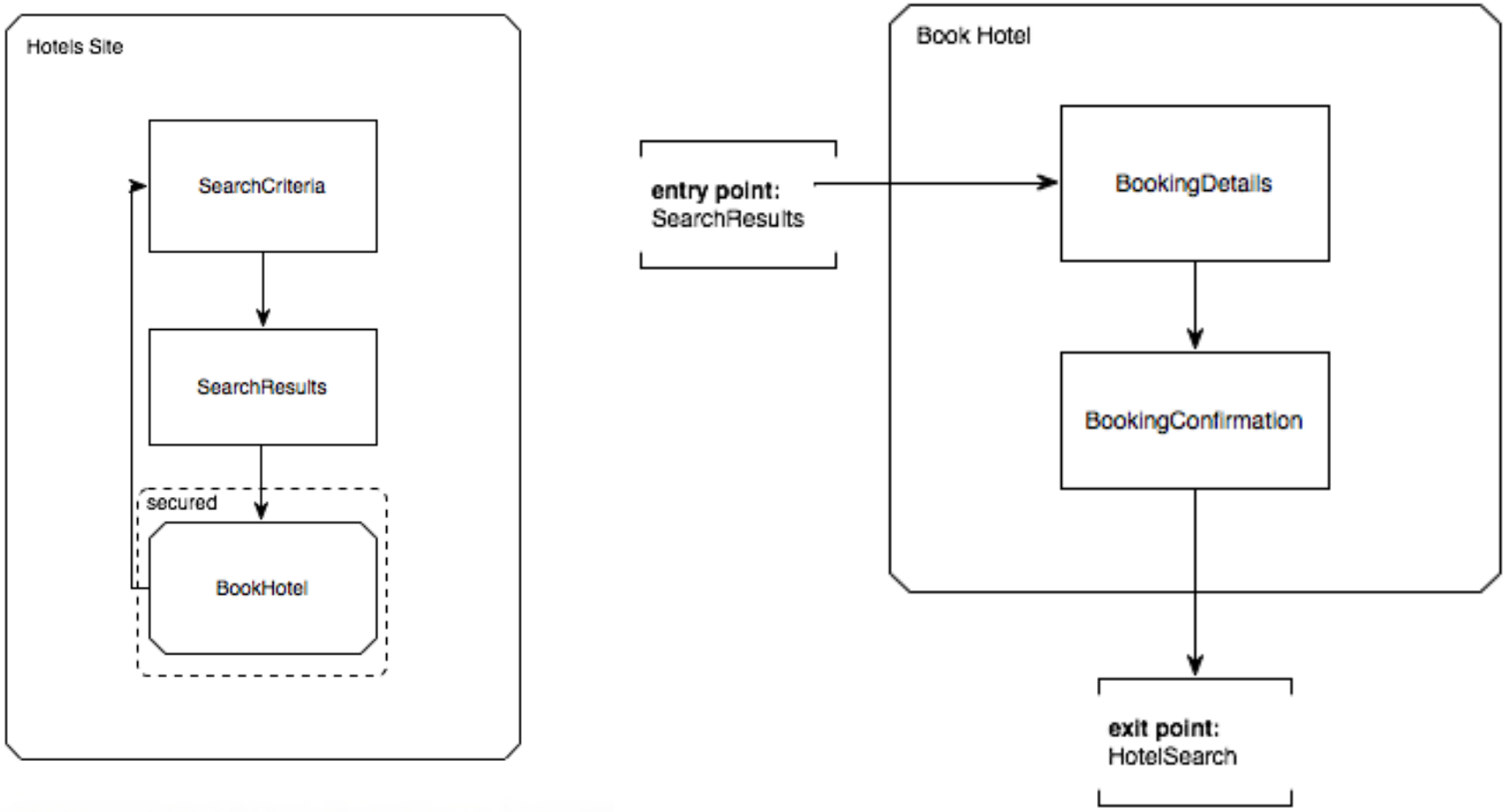
Demo

- Spring Travel Reference Application
  - @Controller usage

Spring Web Flow

- An official Spring MVC extension for implementing flows
  - "A flow encapsulates a reusable sequence of steps that can execute in different contexts"
- Web Flow 2 greatly simplifies the flow programming model
  - SWF 2 flows on average 50% smaller than SWF 1
- Web Flow 2 adds major new features
  - Comprehensive JSF support
  - First class Ajax support
  - Spring Security integration
  - View scope

Web Flow Sweet Spot



Web Flow 2 to Web Flow 1

- Spring Travel Reference App v1
  - 163 lines of Flow code across 3 artifacts
    - Flow definition, Form Action Class, Bean File
- Spring Travel Reference App v2
  - **33 lines of Flow code in 1 artifact**
- Savings attributed to
  - More concise flow definition syntax
  - Enhanced expression language integration
  - Declarative view model binding and validation
  - Also, a new flow inheritance capability can further simplify large applications

Flow Definition Syntax Compared

Invoking Actions

v2

```
<evaluate expression="bookingService.createBooking(hotelId)"
              result="flowScope.booking" />
```

v1

```
<bean-action bean="bookingService" method="createBooking">
    <method-arguments>
        <argument value="${flowScope.hotelId}" />
    </method-arguments>
    <method-result name="booking" scope="flow" />
</bean-action>
```

View Model Binding

v2

```
<view-state id="enterBookingDetails" model="booking">
    <transition on="proceed" to="reviewBooking" />
    <transition on="cancel" to="cancel" bind="false" />
</view-state>
```

v1

```
<view-state id="enterBookingDetails"
          view="enterBookingDetails.jsp">
    <render-actions>
        <action bean="formAction" method="setupForm" />
    </render-actions>
    <transition on="proceed" to="reviewBooking">
        <action bean="formAction" method="bindAndValidate" />

    </transition>
    <transition on="cancel" to="cancel" />
</view-state>
```

Mapping Flow Input/Output

v2

```
<subflow-state id="bookHotel" subflow="booking">
    <input name="hotelId" />
    <transition on="bookingConfirmed" to="finish" />
</subflow-state>
```

v1

```
<subflow-state id="bookHotel" flow="booking">
    <attribute-mapper>
        <input-mapper>
            <mapping source="flowScope.hotelId"
                     target="hotelId"
        </input-mapper>
   </attribute-mapper>
   <transition on="bookingConfirmed" to="finish" />
</subflow-state>
```

Flow Definition Inheritance

parent.xml

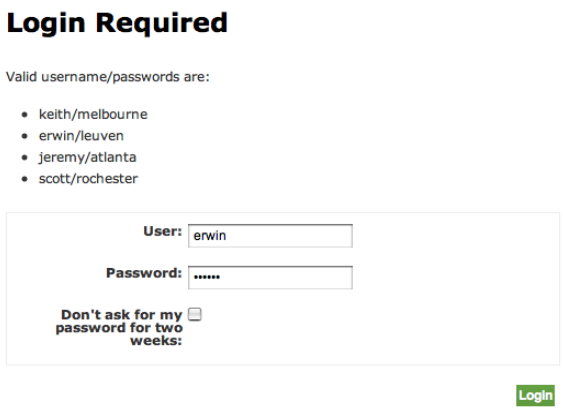
```
<flow abstract="true">
    <!-- Common elements -->
</flow>
```

child.xml

```
<flow parent="parent">
    <!-- Your specific elements -->
</flow>
```

Spring Security Integration

```
<flow>
    <secured attributes="ROLE_USER" />
</flow>
```



- Can also secure states and transitions
- Can refer to user Principal using EL
  - Use the **currentUser** implicit variable
  - Resolvable within flow and view templates

Flow-Managed Persistence

```
<flow>
    <persistence-context/>
</flow>
```

- Scopes a persistence context with this flow
- Automatically used by your data access objects
- Can also call the flow's EntityManager directly using EL
  - e.g. *entityManager.persist(booking)*

Ajax with View Scope

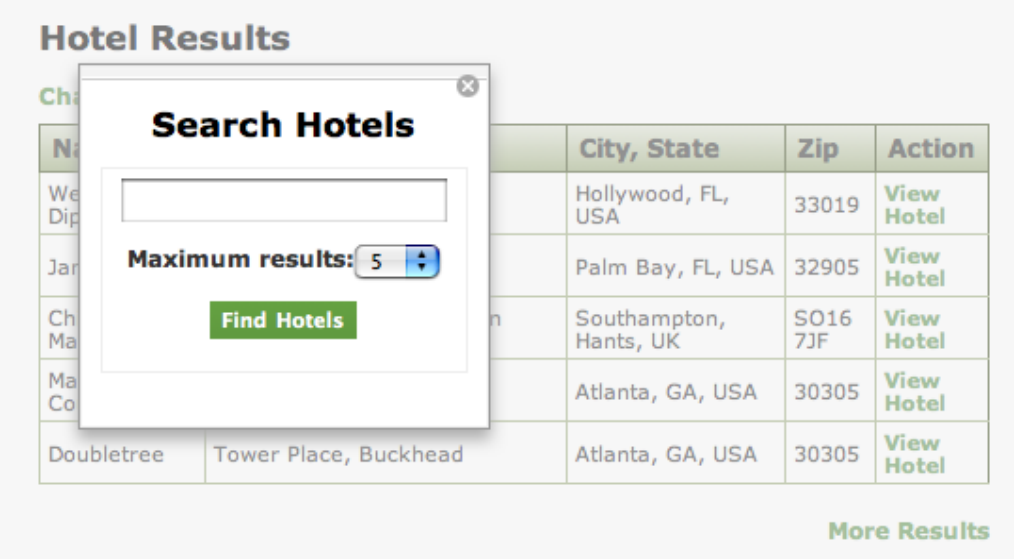
```
<view-state id="reviewHotels">
    <on-render>
        <evaluate expression="service.findHotels(criteria)"
                  result="viewScope.hotels" />
    </on-render>
    <transition on="previous">
        <evaluate expression="criteria.previousPage()" />
        <render fragments="hotelsTable" />
    </transition>
    <transition on="next">
        <evaluate expression="criteria.nextPage()" />
        <render fragments="hotelsTable" />
    </transition>
    <transition on="select" to="reviewHotel" />
    <transition on="changeSearch" to="changeCriteria" />
</view-state>
```

View scope

Partial page rendering

Popups

```
<view-state id="changeCriteria" popup="true">
    <transition on="search" to="reviewHotels" />
</view-state>
```



JSF Support - "Spring Faces"

- Combines the JSF UI component model with Web Flow navigation/state management
  - All in a native Spring MVC environment
- Spring Faces also includes a lightweight JSF component library
  - Designed for the 80%
    - Includes Ajax support, client-side form validation
  - Built on a new Javascript module called "Spring.js"
    - Integrates Dojo as the primary UI toolkit
    - Applies progressive enhancement techniques

Demo

- Spring Travel Reference Application
  - Web Flow usage
  - Spring Faces usage
  - Spring Security integration
  - Progressive enhancement

Summary

- Spring MVC 2.5 is a great leap forward
  - Elegant @Controller model
- Web Flow 2 is also a great leap forward
  - Simplified flow definition language
  - Groundbreaking work in the areas of Progressive Ajax and JSF
- Give them a try!
- Get involved in the Spring community at http://forum.springframework.org
- These slides are also available at http://blog.springsource.com
  - Be sure to subscribe

Questions?

Rossen.Stoyanchev@SpringSource.com