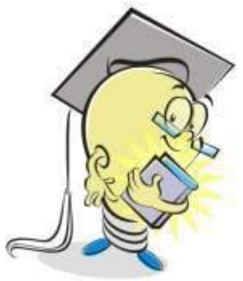
HandlerMapping



C3: Protected

About the Author

Created By:	Ramesh CP (161646(
Credential Information:	SCJP, 8+ years of experience in technical training
Version and Date:	SpringMVC/PPT/1011/3.0

Cognizant Certified Official Curriculum

Icons Used

Questions

Tools

Hands on Exercise

Coding Standards

Test Your Understanding

Reference

Demonstration

A Welcome Break

Contacts

HandlerMapping: Overview

- **Introduction:**
  - Using a handler mapping you can map incoming web requests to appropriate handlers.
  - In previous versions of Spring, users were required to define HandlerMappings in the web application context to map incoming web requests to appropriate handlers. With the introduction of Spring 2.5, the DispatcherServlet enables the DefaultAnnotationHandlerMapping, which looks for @RequestMapping annotations on @Controllers.

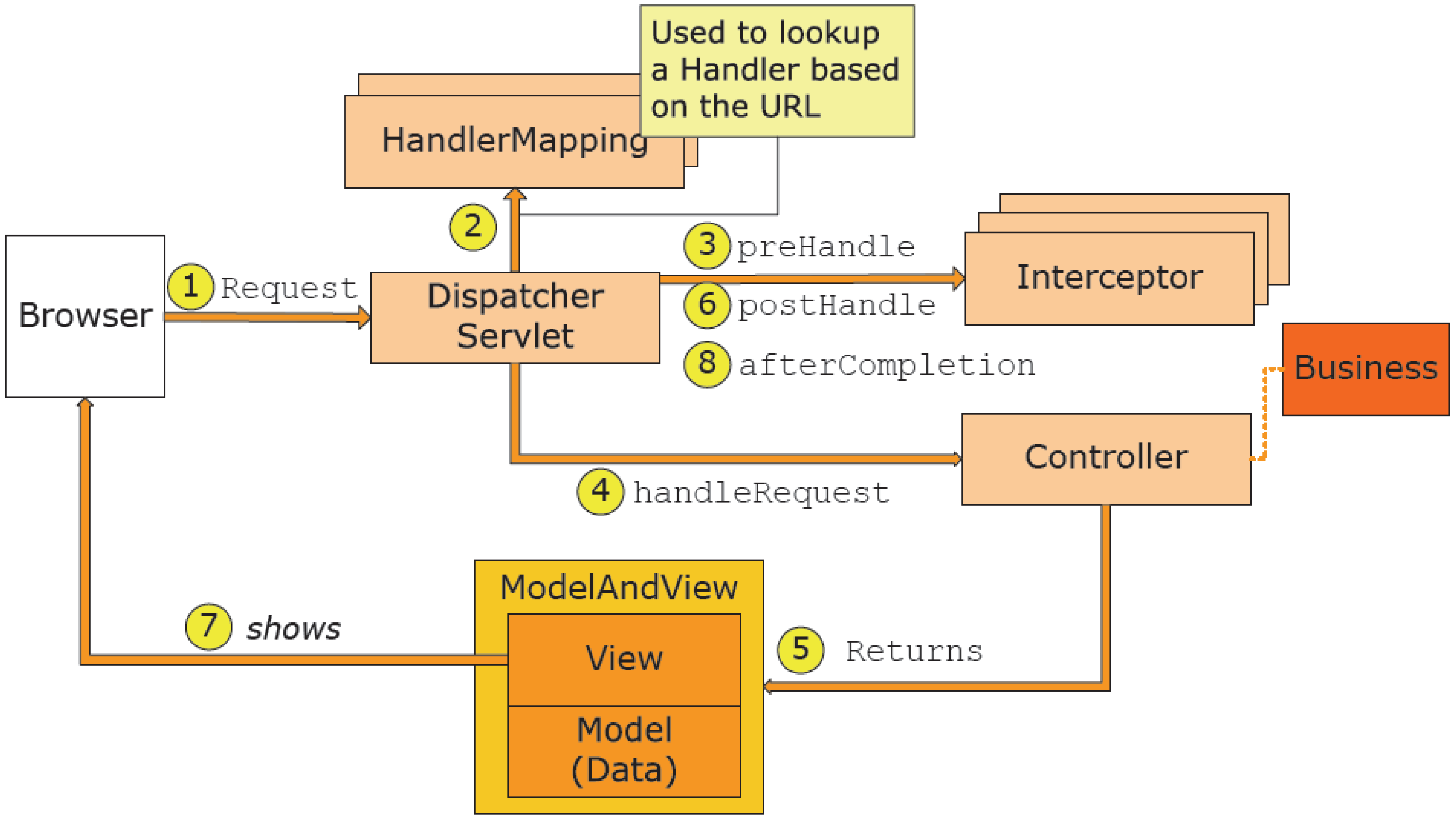
HandlerMapping: Objectives

- Objective:

After completing this chapter you will be able to:

  - Map request to the request handlers (Controllers) using @RequestMapping annotations

Review of Architecture



Mapping URLs to Controllers

- Spring 2.5 provided multiple ways to map URLs (requests) to Spring MVC Controllers
  - BeanNameUrlHandlerMapping (default)
  - SimpleUrlHandlerMapping
  - ControllerClassNameHandlerMapping
  - Annotations
- As of Spring 3.0, the recommended approach for using controllers is to favor annotations
  - More on this approach later

Using Handler Mappings

- A `HandlerMapping` implementation returns a Handler based on the `Request`
  - A Handler chain consists of a `HandlerInterceptor` and the actual handler
  - The `HandlerInterceptor` consists of three methods:

`preHandle`	request, response, handler (Called as a before interceptor)
`postHandle`	request, response, handler, modelAndView (Called as an after interceptor)
`afterCompletion`	request, response, handler, Exception (Called after the view has been rendered)

Using Handler Mappings (Contd.)

❖ Standard implementations:

`BeanNameUrlHandlerMapping` (Default)	Maps URLs to bean ids (or aliases) in the WebApplicationContext (similar to action mapping in Struts).
`SimpleUrlHandlerMapping`	Uses a mapping between URL patterns and controllers. Very flexible, as it allows wildcards.
`CommonsPathMapHandlerMapping`	Deprecated in Spring 2.5 in favor of request-based annotation mapping.

Handler Mapping Examples

❖ Example of the **`BeanNameUrlHandlerMapping:`**

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN"
        "http://www.springframework.org/dtd/spring-beans.dtd">
<beans>
  <!-- Don't need to specify the BeanNameUrlHandlerMapping as
       it is the default mapping used -->
  <bean name="/Register.htm"
        class="demo.springmvc.RegistrationController"/>
  <bean name="/Subscribe.do"
       class="demo.springmvc.newsletter.SubscribeController"/>
  <!-- other beans -->
</beans>
```

Handler Mapping Examples (Contd.)

❖ Example using **`SimpleUrlHandlerMapping`**:

```
<bean id="handlerMapping"
      class="....web.servlet.handler.SimpleUrlHandlerMapping">
  <property name="mappings">
    <props>
      <prop key="/catalog/*.do">catalogController</prop>
      <prop key="/register/Registration.form">
        regController
      </prop>
    </props>
  </property>
</bean>
<bean id="catalogController" .../>
<bean id="regController" ...>
```

Handler Mapping Examples (Contd.)

❖ The order can be specified using the `order` property

```
<bean id="handlerMapping" ...>
  <property name="order">
    <value>0</value>
  </property>
  ...
</bean>
<bean id="handlerMapping2" ...>
  <property name="order">
    <value>1</value>
  </property>
  ...
</bean>
```

HandlerMapping Interface

- The HandlerMapping interface defines a mapping between requests and handler objects.

```
package org.springframework.web.servlet;
public interface HandlerMapping{
HandlerExecutionChain getHandler(HttpServletRequest request)
        throws Exception;
}
```

- DefaultAnnotationHandlerMapping implements HandlerMapping interface that maps handlers based on HTTP paths expressed through the RequestMapping annotation at the type or method level.

Annotation Driven Configuration

- You only need to add a single line of configuration to spring configuration xml file to flip on all of the annotation-driven features you'll need from Spring MVC.

  ```
  <mvc:annotation-driven/>
  ```

- To configure Spring for autodiscovery, use <context:component-scan>. The <context:component-scan> element works by scanning a package and all of its subpackages, looking for classes that could be automatically registered as beans in the Spring container. The base-package attribute tells <context:component-scan> the package to start its scan from.

  ```
  <context:component-scan base-package="com.springinaction.springidol">
  </context:component-scan>
  ```

- Annotation-driven and autodiscovery scan can dramatically reduce the amount of XML Spring configuration. You'll need only a handful of lines (as above) of XML, regardless of how many beans are in your Spring application context.

Spring XML Configuration Example

```
<?xml version="1.0" encoding="UTF-8"?> <beans
    xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-
        3.0.xsd http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context-
        3.0.xsd" >
```

The simplest Spring XML configuration

```
<mvc:annotation-driven/>
<context:component-scan base-package="com.springinaction.springidol">
    </context:component-scan>


</beans>
```

<mvc:annotation-driven>

- Sets up defaults within an MVC app
  - Establishes the DefaultAnnotationHandlerMapping bean
  - Establishes AnnotationMethodHandlerAdaptor as the default HandlerAdaptor
    - This defines how to map URLs to handler methods
  - Adds support for @RequestParam
  - Establishes default formatting converters and validators

Handling Requests (Annotations)

- Annotation-based mapping (Spring 2.5 or later)
  - Uses the @Controller annotation to define controllers
  - Autodetected through classpath scanning
  - Usually used in conjunction with @RequestMapping annotation

RequestMapping Annotation

- RequestMapping annotation are used for mapping web requests onto specific handler classes and/or handler methods. In other words, the annotation defines mapping rules.

package org.springframework.web.bind.annotation

```
@Target(value={METHOD,TYPE})
@Retention(value=RUNTIME)
@Documented
public @interface RequestMapping
```

Handler Methods Parameters

- Methods annotated using @RequestMapping may have very flexible signatures
  - Methods may accept as input parameters:
  - Request or response objects
  - Session object
  - Command objects (e.g. Employee, Product, etc.)
  - @PathVariable, @RequestParam, @RequestBody annotated parameters
  - Errors or BindingResult objects

RequestMapping at class level

- You use the @RequestMapping annotation to map URLs such as /appointments onto an entire class or a particular handler method.
- Typically the class-level annotation maps a specific request path (or path pattern) onto a form controller.

Mapping Requests

- By path at class level and/or method level
  - @RequestMapping(“path”)
- By HTTP method
  - @RequestMapping(“path”, method = RequestMethod.Get)
  - POST, PUT, DELETE, OPTIONS, and TRACE are also supported.
- By Presence of query parameter
  - @RequestMapping(“path”, method = RequestMethod.GET, params=“foo”)
- By Presence of request header
  - @RequestMapping(“path”, header = “content-type=text/*”)

RequestMapping – Method level

- The RequestMapping annotation serves two purposes.
- First, it identifies showHome() as a request-handling method. And, more specifically, it specifies that this method should handle requests whose path is /home.
- Method-level @RequestMappings narrow the mapping defined by any class-level @RequestMapping.

```
@Controller
Public class HomeController {
    @RequestMapping("/home")
    public String showHome() {
            return "hello_world";
    }
}
```

RequestMapping – class level

- @RequestMapping can be used at the class level. It is concise way to map all requests within a path to a @Controller.

```
@Controller
@RequestMapping(“/appointments/*”)
public class AppointmentsController {
    @RequestMapping(“active”)
    public List<Appointment> active() { }

    @RequestMapping(“inactive”)
    public List<Appointment> inactive() {}
}
```

RequestMapping – class level (Contd.)

❖ The same mapping rules can be applied at the method level also.

```
@Controller
public class AppointmentsController {
    @RequestMapping(“/appointments/active”)
    public List<Appointment> active() { }

    @RequestMapping(“/appointments/inactive”)
    public List<Appointment> inactive() {}
}
```

RequestMapping – HTTP GET

- ❖ @RequestMapping annotation can be applied at method levels primary mapping for a specific HTTP method request method ("GET"/"POST")
- ❖ The following queryAppointment only accepts GET requests, meaning that an HTTP GET for /appointments invokes this method.

```
@Controller
@RequestMapping(“/appointments/*”)
public class AppointmentsController {
    @RequestMapping(value=“query”, method=GET)
    public String queryAppointment(@RequestParam(“appointmentId”) String
    appointmentId)  {
        /* logic to get specific appointment detailes  */
      model.addAttribute(appointment);
        return “appointmentDetails”;
    }
}
```

Request Mapping – HTTP Post

- AddAppointment() method has a further @RequestMapping refinement and it handles only POST requests.

```
@Controller
@RequestMapping("/appointments/")
public class AppointmentsController {
@RequestMapping(method=RequestMethod.POST)
    public String addAppointment(@Valid Appointment appointment) {
            //logic for saving appointment in database
            return "success";
    }
}
```

Custom Handler Mapping

- If you define custom HandlerMapping beans in your DispatcherServlet context, you need to add a DefaultAnnotationHandlerMapping bean explicitly.
- Custom HandlerMapping beans replace the default mapping strategies.
- Following is the example of defining a DefaultAnnotationHandlerMapping for registering custom interceptors:

```
<bean
    class="org.springframework.web.servlet.mvc.annotation.DefaultAnnotatio
    nHandlerMapping">
  <property name="interceptors">
   ...
  </property>
</bean>
```

HandlerMapping

Time for a Break !

HandlerMapping

- Questions from participants

Test Your Understanding

1) Identify the class from the following you need to add in spring config xml for custom handling :

   a) BeanNameUrlHandlerMapping

   b) SimpeUrlHandlerMapping

   c) DefaultAnnotationHandlerMapping

   d) None of the above

2) RequestMapping annotation can be applied at class and/or method level. State true or false.

HandlerMapping: Summary

- With the help of @RequestMapping annotation you can map incoming web requests to appropriate handlers.
- @ RequestMapping annotation defines mapping rules at class and/or method levels.
- If you need to add DefaultAnnotationHandlerMapping bean explicitly for custom HandlerMapping in your DispatcherServlet context.

HandlerMapping: Source

- http://static.springsource.org/spring/docs/3.0.x/spring-framework-reference/html/mvc.html#mvc-ann-requestmapping

You have successfully completed HandlerMapping

Click here to proceed

Cognizant™
Passion for building stronger businesses