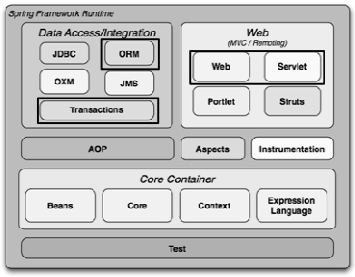
CS520 Web Programming

Spring – Web MVC

Chengyu Sun
California State University, Los Angeles

Spring Framework



Roadmap

- Introduction to Spring MVC
- Database access
- Controller examples
- Spring MVC in CSNS2

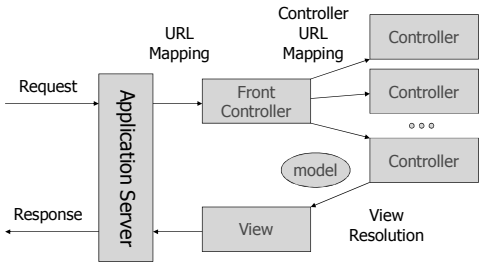
The SHAM Example

- Spring and Hibernate from Scratch
  - http://csns.calstatela.edu/wiki/content/cysun/course_materials/cs520/sham/

Add Spring MVC to A Maven Webapp

- Spring dependencies
  - `spring-webmvc`
- Front controller
  - `DispatcherServlet` in web.xml
- Bean configuration file
  - /WEB-INF/<servlet-name>-servlet.xml

Request Processing in Spring Web MVC



Spring Controllers

- Spring controllers are *beans* annotated with `@Controller`
- In <servlet-name>-servlet.xml
  - `<mvc:annotation-driven>`
  - `<context:component-scan>`

Controller URL Mapping

- Requests are mapped to controller methods using `@RequestMapping`
  - `value`: URL pattern(s)
  - Mapping can be further refined by using `method`, `params`, and `headers`
  - http://static.springsource.org/spring/docs/current/spring-framework-reference/html/mvc.html#mvc-ann-requestmapping

View

- A controller method returns a view name, which will be resolved to a view implementation by a view resolver

Resolve JSP Views

```
<bean id="viewResolver"
   class="org.springframework.web.servlet.view.InternalResourceViewResolver">
      <property name="prefix" value="/WEB-INF/jsp/" />
      <property name="suffix" value=".jsp" />
</bean>
```

Prefix + ViewName + Suffix = JSP File

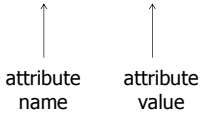
Why not just return the path to the JSP file??

View Technologies and Resolvers

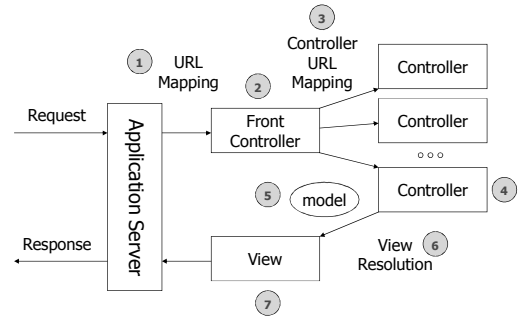
- http://static.springsource.org/spring/docs/current/javadoc-api/org/springframework/web/servlet/ViewResolver.html

Model

- Model objects are passed to view using a `ModelMap`
  - `put( String, Object )`



Request Processing Recap



Database Access Dependencies

- Hibernate
- Spring ORM
- Database connection pooling
  - DBCP vs. Tomcat JDBC
- Database driver

Configuration

- Hibernate JPA
  - persistence.xml
- Spring
  - web.xml
  - applicationContext.xml

Spring Transaction Management

- Transaction management code is added through AOP
- <context:annotation-config> enables annotations like `@Autowired` and `@PersistenceContext`
- <tx:annotation-driven> enables annotations like `@Transactional`

Model Classes and Database

- Model classes, e.g. `User`
  - JPA annotations
- Database
  - SQL scripts
  - `hibernate_sequence`

Database Access

- DAO interfaces, e.g. `UserDao`
- DAO implementations, e.g. `UserDaoImpl`

Spring DAO Annotations

- @Repository for DAO implementation classes
- @PersistenceContext injects an entity manager to the class
- @Transactional for methods that write to the database

Controller Examples

- List all users
- Display a selected user
- Add a new user
- Edit a user

Example: List All Users

- Controller basics
  - `@Controller`
  - `@RequestMapping`
  - `@ModelMap`
  - Using DAO

Example: Display A User

- Using `@RequestParam`
  - `required`
- Using `@PathVariable`

About Wildcard in Servlet <url-pattern>

- A string beginning with a ***.**
  - E.g. `*.html`, `*.do`
- A string beginning with a **/** and ending with a **/***
  - E.g. `/*`, `/users/*`

See Servlet Specification 3.0, Section 12

About Spring @RequestMapping

http://localhost/springmvc/user/viewUsers.html

⇩

<url-pattern>*.html</url-pattern>

⇩

@RequestMapping("/usr/viewUsers.html")

http://localhost/springmvc/users/1

⇩

<url-pattern>/users/*</url-pattern>

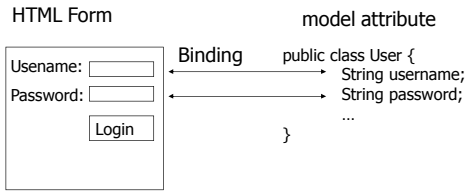
⇩

@RequestMapping("/{userId}")

Example: Add A User

- Model attribute (a.k.a. command object, form backing object)
  - Concept
  - Binding
  - `@ModelAttribute`
- `@RequestMapping`
  - `method`
- The "redirect" view

Command Object



- Bind form fields to properties of the object

Handling Forms

- One controller method for
  - Creating a model attribute
  - Returning the form view
- Form view
  - Use Spring `<form>` tags to bind the model attribute to the form
- One controller method for
  - Processing the command object (annotated with `@ModelAttribute`)
  - Returning the success view

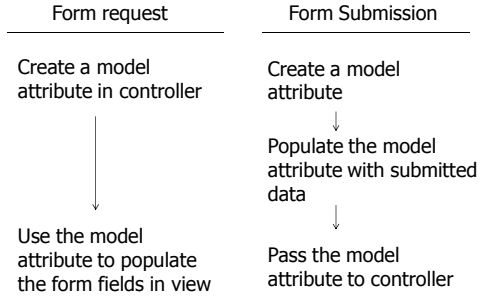
Spring's `form` Tag Library

- Documentation - http://static.springsource.org/spring/docs/current/spring-framework-reference/html/view.html#view-jsp-formtaglib
- Tag reference – http://static.springsource.org/spring/docs/current/spring-framework-reference/html/spring-form.tld.html

Example: Edit A User

- Store model attribute in session
  - `@SessionAttributes`
  - `SessionStatus`
    - Use `setComplete()` to remove the model attribute in session

Non-Session Model Attributes



Session Model Attributes

Form request	Form Submission
Create a model attribute in controller	Retrieve the model attribute from session
↓	↓
Save the model attribute in session	Populate the model attribute with submitted data
↓	↓
Use the model attribute to populate the form fields in view	Pass the model attribute to controller

Access HTTP Request, Response, and Session

- Simply add an argument of that type to the controller method
- Examples in CSNS2
  - DownloadController
  - SectionController

Spring MVC in CSNS2

- Configuration
- Apache Tiles

Additional Spring Configurations

- web.xml
  - URL pattern for the front controller
  - `OpenEntityManagerInViewFilter`
- csns-servlet.xml
  - `<mvc:resources>`
  - `<mvc:view-controller>`

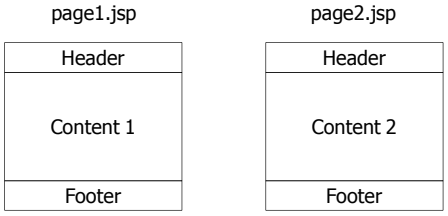
Resource Filtering Using Maven

- Replace place holders like `${db.url}` in resource files with actual values
- `<filter>`
  - build.properties
- Resources
  - `<resources>`, `<testResources>`, `<webResources>`

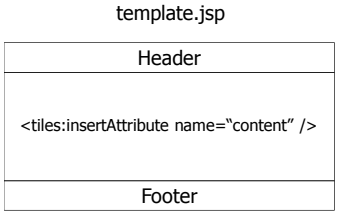
Apache Tiles

- http://tiles.apache.org/
- A template framework for JSP
- Similar to Master Page in ASP.NET

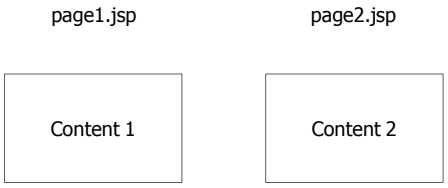
A Tiles Example



Template Page



Content Pages



Definitions

```
<definition name="page1" template="template.jsp">
   <put-attribute name="content" value="page1.jsp" />
</definition>

<definition name="page2" template="template.jsp">
   <put-attribute name="content" value="page2.jsp" />
</definition>
```

Definition Extension

- Like in an OO language, a Tiles definition can *inherits* from another definition, and *overrides* some attributes in the parent page

```
<definition name="page2" extends="page1">
   <put-attribute name="content" value="page2.jsp" />
</definition>
```

Resolve Tiles Views in Spring

```
<bean class="org.springframework.web.servlet.view.tiles2.TilesViewResolver"/>

<bean class="org.springframework.web.servlet.view.tiles2.TilesConfigurer">
   <property name="definitions">
      <list>
         <value>/WEB-INF/tiles.xml</value>
      </list>
   </property>
</bean>
```

Tiles Usage in CSNS2

- `WEB-INF/layouts` for template and header/footer pages
- `WEB-INF/content` for content pages
- `WEB-INF/tiles.xml`