CS520 Web Programming
Spring – MVC Framework

Chengyu Sun
California State University, Los Angeles

Roadmap

- Request Processing
  - DAO implementation (Model)
  - Tiles (View)
- Controllers (Controller)

Understand Request Processing

- What happens when the server received a request like

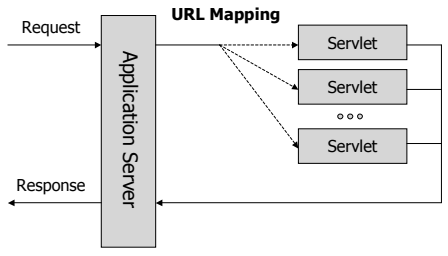
```
http://localhost:8080/csns2/student/viewSubmission.html?assignmentId=2000005
```

Host URL — Application Context Name

Application Context

- Allows multiple applications to be deployed on the server without interfering with each other
- Application context name
  - Project name in Eclipse, or
  - WAR file name, or
  - Specified in `WEB-INF/META-INFO/context.xml`

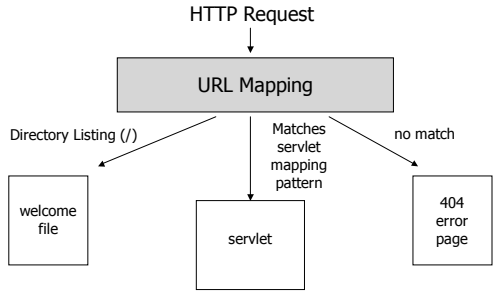
URL Mapping in a Java EE Application



URL Mapping …

- Configured in web.xml
  - <servlet> and <servlet-mapping>
  - <welcome-file-list>
  - <error-page>

… URL Mapping



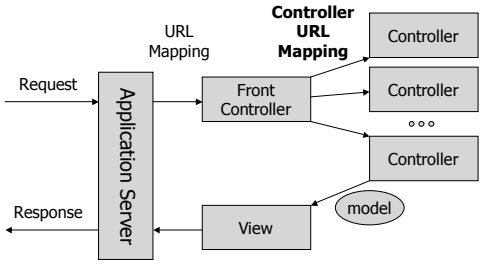
Other Configuration Files

- Under classpath
  - Properties files
  - Hibernate, EHCache, Log4j, …
- Under WEB-INF
  - Spring bean definitions
  - Tiles, …

Spring MVC Namespace

- <mvc:annotation-driven>
- <mvc:resources>
  - Serve static resources
  - Can set cache period
- <mvc:view-controller>
  - A controller that simply returns a view (without doing anything else)

Request Processing in an MVC Framework



Controller URL Mapping

- Controller classes are annotated with @Controller
- Controller methods are annotated with @RequestMapping
  - `value`: URL pattern(s)
  - Mapping can be further refined by using `method`, `params`, and `headers`

Database Access

- DAO interfaces
  - E.g. `csns.model.core.dao`
- DAO implementations
  - E.g. `csns.model.core.dao.jpa`

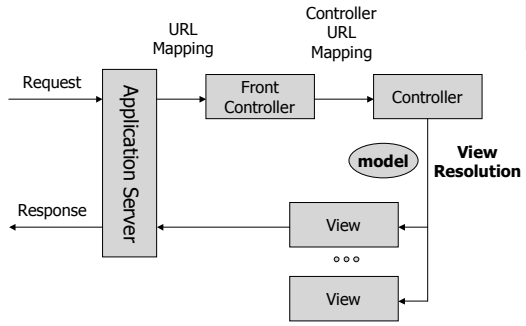
Spring Transaction Management

- Transaction management code is added through AOP
- <tx:annotation-driven> enables configuration of transactional behavior based on annotations

Spring DAO Annotations

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

Request Processing in an MVC Framework



Model and View

- A view is identified by a *name* (not a path like /WEB-INF/foo.jsp)
- Model objects are passed to view using a `ModelMap`

View Technologies Supported by Spring

- JSP, Velocity, FreeMarker, XSLT, Tiles, JasperReports
- Views generated by Java classes

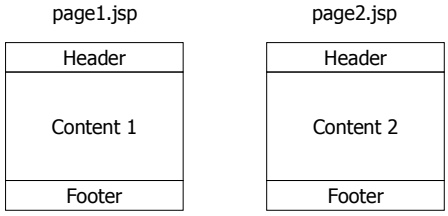
View Resolvers

- http://static.springsource.org/spring/docs/3.0.x/javadoc-api/org/springframework/web/servlet/ViewResolver.html

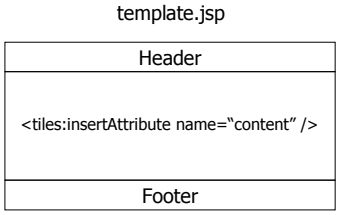
Apache Tiles

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

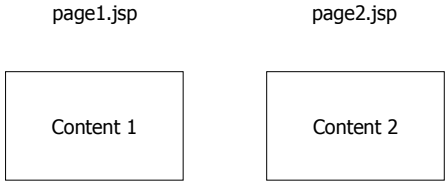
A Tiles Example



Tiles – Template Page



Tiles – Content Pages



Tiles – 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>
```

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>
```

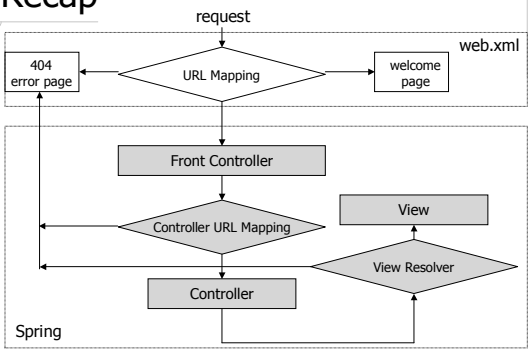
Tiles Usage in CSNS2

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

Special Views

- "`redirect:`" + url
  - E.g. instructor/AssignmentController
- `null`
  - E.g. DownloadController

Recap



Create Controllers

- List all courses
- Display a selected course
- Add a new course
- Edit a course

Example: List All Courses

- Controller basics
  - `@Controller`
  - `@RequestMapping`
  - `ModelMap`
  - Database access
- Use Tiles
  - Create a content page
  - Add a tiles definition

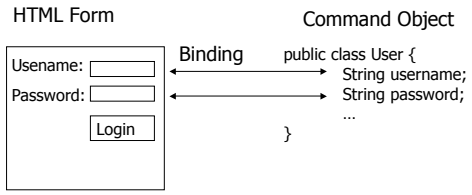
Example: Display A Course

- `@RequestParam`
  - `required`

Example: Add A Course

- Command object (a.k.a. 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 command object
  - Returning the form view
- Form view
  - Use Spring `<form>` tags to bind the command object 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 Course

- Store command object in session
  - `@SessionAttributes`
  - `SessionStatus`
- Property editor
  - Convert between an object and its string representation
  - E.g. a `Calendar` object ⇔ "10/26/2011"
- The `<select>` tag in Spring `form` taglib

Exercise: Delete A Course

- Delete the course from the database, or
- Set a `deleted` flag

Access HTTP Request, Response, and Session

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

Further Readings

- Spring in Action (3rd Ed)
  - Chapter 5-7
- Spring Framework Reference Documentation http://static.springsource.org/spring/docs/current/spring-framework-reference/html/
  - Chapter 16.2.4 Using Spring's form tag library
  - Appendix G. spring-form.tld