CS520 Web Programming
Spring – MVC Framework

Chengyu Sun
California State University, Los Angeles

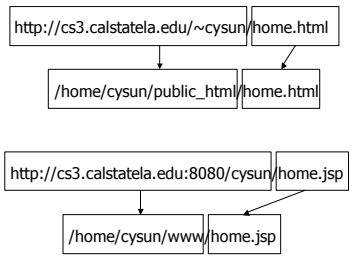
Roadmap

- **Web flow**
- Controllers and validation
- Transactions and hibernate support
- Bits and pieces
  - Logging
  - Context, data source, and connection pooling
  - Creating WAR files
  - JSP pre-compilation
  - Displaytag

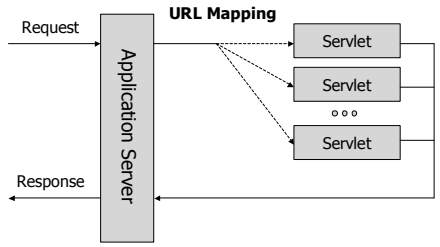
Understand Web Flow

- What happens when the server received a request like `http://cs3.calstatela.edu:8080/csns/instructor/home.html`

Direct Resource Mapping



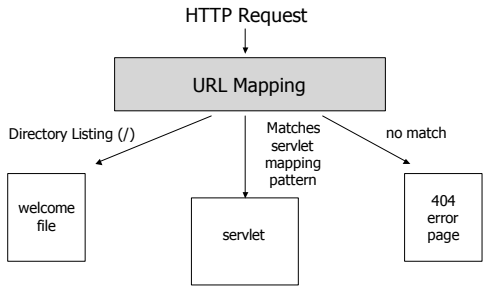
Web Flow of Simple J2EE Application



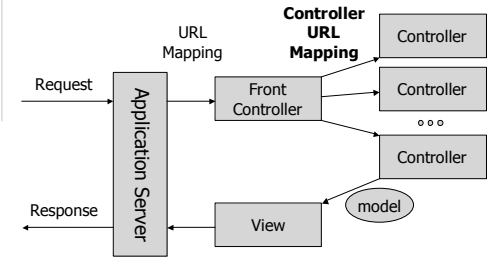
URL Mapping …

- Specified by the Servlet Specification
- Configured in web.xml
  - <servlet> and <servlet-mapping>
    - "`/servlet/*`"
    - "`*.do`"
  - <welcome-file-list>
  - <error-page>

… URL Mapping



Web Flow of a MVC Framework



Spring Configuration File(s)

- <name>-servlet.xml
  - <name> must be the same as the <servlet-name> of the DispatcherServlet specified in web.xml
- Optional
  - csns-data.xml
  - csns-service.xml

More About Configuration Files (Welcome to Metadata Hell!)

- Under classpath (/WEB-INF/classes)
  - hibernate.cfg.xml
  - ehcache.xml
  - *.properties
- Under /WEB-INF
  - web.xml
  - csns-servlet.xml, csns-data.xml, csns-service.xml
  - server-config.wsdd
- Under /META-INF
  - context.xml

Controller URL Mapping …

- Maps a URL pattern to a controller that will handle the request

BeanNameUrlHandlerMapping (default)

```
<bean id="index" name="/index.html /public/*index.html"
      class="evelyn.spring.controller.IndexController" />
```

… Controller URL Mapping …

SimpleUrlHandlerMapping

```
<bean id="index" class="evelyn.spring.controller.IndexController" />

<bean id="urlMapping"
      class="org.springframework.web.servlet.handler.SimpleUrlHandlerMapping">
    <property name="mappings">
      <props>
        <prop key="/index.html">index</prop>
        <prop key="/public/*index.html">index</prop>
      </props>
    </property>
  </bean>
```

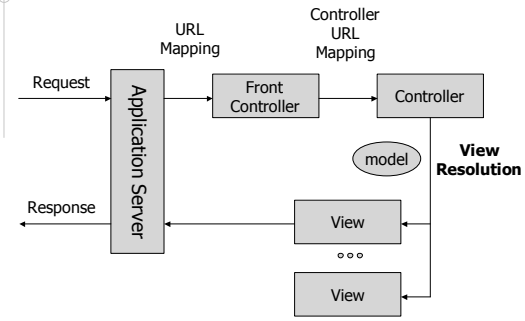
… Controller URL Mapping

- More than one URL handler
  - <property name="order" value="1"/>
- No mapping found
  - 404 error

ModelAndView

ModelAndView (
String viewName, → Resolve to a view
String modelName, → Attribute in Request scope
Object modelObject
)

Web Flow of a MVC Framework



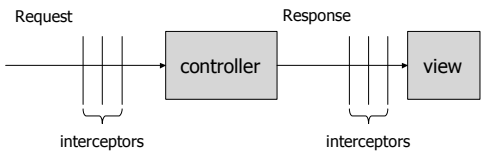
View Resolvers

- URL-based resolvers
  - `InternalResourceViewResolver` for JSP
  - `VelocityViewResolver` for Velocity
- View classes resolvers
  - `BeanNameViewResolver`
  - `XmlViewResolver`
  - `ResourceBundleViewResolver`
- Multiple resolvers
  - <property name="order" value="1"/>

Special Views

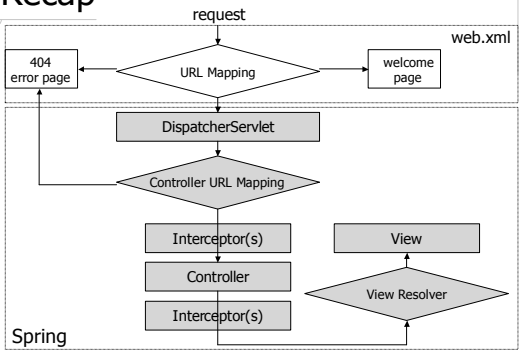
- Redirect
  - new ModelAndView("**redirect:**" + url)
- Forward
  - new ModelAndView("**forward:**" + url)

Interceptors



- Similar to Filters in J2EE specification
- `HandlerInterceptorAdapter`
- Example: AuthorizationInterceptor

Recap



Roadmap

- Web flow
- **Controllers and validation**
- Transactions and hibernate support
- Bits and pieces
  - Logging
  - Context, data source, and connection pooling
  - Creating WAR files
  - JSP pre-compilation
  - Displaytag

Controller Interface

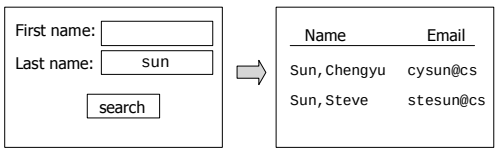
- org.springframework.web.servlet.mvc.Controller - http://www.springframework.org/docs/api/org/springframework/web/servlet/mvc/Controller.html

Select A Controller

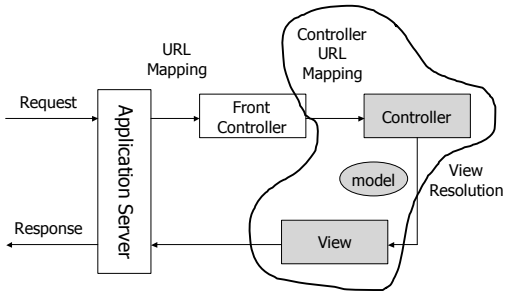
ParameterizableViewController	Simply display a view, i.e. the request does not need to be processed
Controller (interface) AbstractController	do not use request parameters or use only simple parameters
BaseCommandController AbstractCommandController	request parameters can be mapped to an object; can use *validators* to validate request parameters
AbstractFormController SimpleFormController	Handles form input
AbstractWizardFormController	Handles multi-page form input

Example: Find Email by Name

- Given the first name and/or the last name of a user, display the user's email address



So What Do We Need to Do?



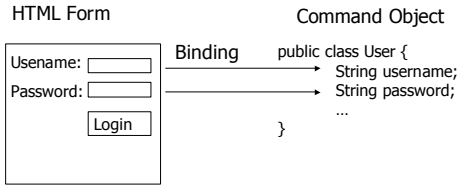
Models and Views

- Models
  - UserDao.getUsersByName() → List<User>
- Views
  - searchEmail.jsp
  - displayEmail.jsp

Controllers

- ParameterizableViewController + AbstractController
- SimpleFormController

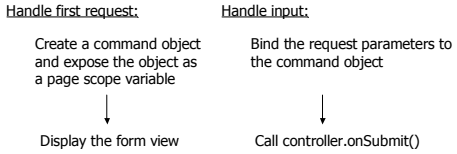
Command Object



- Any class can be used as a command class in Spring
  - vs. ActionForm in Struts

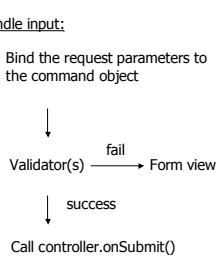
Simple Form Handling

- The controller needs to handle two cases:
  - Display form
  - Process input data



Validation

- org.springframework.validation
  - Validator
  - Errors



messages.properties

- <name,value> pairs
- A single place for output messages
  - Easy to change
  - I18N
- Need to declare a `messageSource` bean in Spring configuration file
- Can be used by <fmt> tags in JSTL

Display Errors

- Spring Tag Libraries
  - <spring> - http://static.springframework.org/spring/docs/2.0.x/reference/spring.tld.html
  - <spring-form> - http://static.springframework.org/spring/docs/2.0.x/reference/spring-form.tld.html
- Example: account.jsp

Limitation of Spring Validation

- *Server-side only*
- Takes lots of coding for anything other than empty/white spaces
- Vs. Commons-validator support in Struts
- Spring Modules - https://springmodules.dev.java.net/

Commons-Validator Framework Example

```
<form name="fooForm">

    <field property="name" depends="required">
        <arg0 key="fooForm.definition"/>
    </field>

    <field property="difficultyLevel"
           depends="required, integer">
        <arg0 key="fooForm.difficultyLevel"/>
    </field>

</form>
```

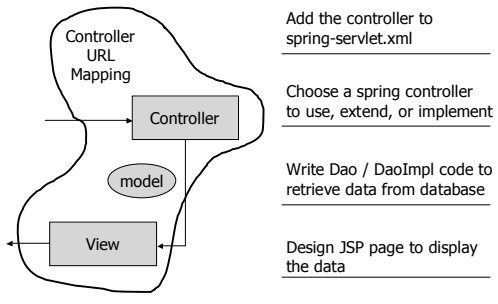
Commons-Validator Routines

- Since 1.3.0
- Independent of Commons-Validator Framework
- A set of methods to validate
  - Date and time
  - Numeric values
  - Currency
  - ISBN
  - Regular expression …

Exception Handling

- An *Exception resolver* catches all exceptions thrown by controllers and chooses the proper view to display
- Examples
  - SimpleMappingExceptionResolver
  - ExceptionResolver in CSNS

Recap



Roadmap

- Web flow
- Controllers and validation
- **Transactions and hibernate support**
- Bits and pieces
  - Logging
  - Context, data source, and connection pooling
  - Creating WAR files
  - JSP pre-compilation
  - Displaytag

Programmatic vs. Declarative Transaction Management

Programmatic:

```
void saveUser( User u )
{
  transaction.start();
  …
  transaction.end();
}
```

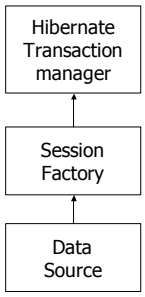
Declarative:

```
<transaction>
   <method>
      saveUser
   </method>
</transaction>
```

Spring Transaction Managers

- JDBC
- Hibernate
  - V3
  - Before V3
- JTA
- Object-Relational Bridge (ORB)

Configure Hibernate Transaction Manager



Transaction Attributes

- Propagation behaviors
- Isolation levels
- Read-only hints
- Transaction timeout period

Propagation Behaviors

- Determines whether the method should be run in a transaction, and if so, whether it should run within an existing transaction, a new transaction, or a nested transaction within an existing transaction.

Adding Transaction Aspect

- TransactionProxyFactoryBean
  - target
  - transactionManager
  - transactionAttributeSource

Other Hibernate Support

- HibernateTemplate
- OpenSessionInViewFilter and OpenSessionInViewInterceptor

Access Hibernate Session Directly in HibernateTemplate

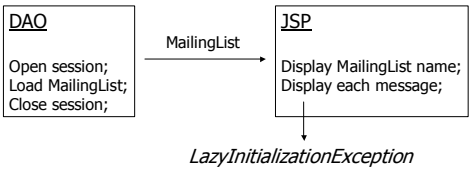
- getSessionFactory().getCurrentSession()
  - For versions of Hibernate before 3.0.1, the transactions are not managed by Spring if the session is obtained this way
- HibernateCallback
  - Object execute(HibernateCallback action)
  - List executeFind(HibernateCallback action)

Hibernate Callback Example

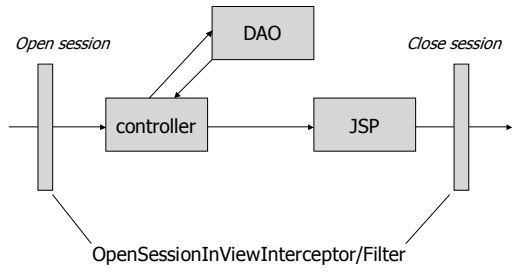
```
public Feedback getFeedback( final Integer itemId,
                             final Integer userId )
{
  return (Feedback) getHibernateTemplate().execute(
    new HibernateCallback() {

      public Object doInHibernate( Session session )
                  throws HibernateException, SQLException
      {
        String hql = "from Feedback f where "
             + "f.item.id = :itemId and f.user.id = :userId";
        Query query = session.createQuery( hql );
        query.setInteger( "itemId", itemId );
        query.setInteger( "userId", userId );
        return query.uniqueResult();
      }

  } );
}
```

Hibernate LazyInitializationException

```
class MailingList {
   String name;
   List<Message> messages;
}
```



Open Session in View



Roadmap

- Web flow
- Controllers and validation
- Transactions and hibernate support
- **Bits and pieces**
  - Logging
  - Context, data source, and connection pooling
  - Creating WAR files
  - JSP pre-compilation
  - Displaytag

Logging

- Use print statements to assist debugging
  - Why do we want to do that when we have GUI debugger??

```
public void foo()
{
    System.out.println( "loop started" );
    // some code that might get into infinite loop
    ...
    System.out.println( "loop finished" );
}
```

Requirements of Good Logging Tools

- Minimize performance penalty
- Support different log output
  - Console, file, database, …
- Support different message levels
  - Fatal, error, warn, info, debug, trace
- Easy configuration

Log4j and Commons-logging

- Log4j
  - A logging tool for Java
  - http://logging.apache.org/log4j/docs/
- Commons-logging
  - A wrapper around different logging implementations to provide a consistent API
  - http://jakarta.apache.org/commons/logging/

Log4j Configuration File

- *log4j.properties* or *log4j.xml*
- Appender
  - Output type
  - Output format
- Logger
  - Class
  - Message level

Log4j PatternLayout

- http://logging.apache.org/log4j/docs/api/org/apache/log4j/PatternLayout.html

Log4j Example

Connection Pooling

- Connection pooling
- DBCP
  - http://jakarta.apache.org/commons/dbcp/configuration.html

Configure Connection Pooling

- At the application server level
  - Provided through JNDI
  - E.g. http://tomcat.apache.org/tomcat-5.5-doc/jndi-datasource-examples-howto.html
- At the application level
  - E.g. <bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource" />

Creating WAR Files

- Understand the directory structure
- Use the war ANT task

JSP Pre-compilation

- Usually a JSP is converted to a servlet and then compiled into byte code *when the JSP is request for the first time*.
- JSP pre-compilation
  - Eliminate the "first request overhead"
  - Speed up development
- Tomcat provides JSP pre-compiler which can be used as an ANT task
  - Need to specify the compiled servlets in web.xml

Displaytag

- http://displaytag.sourceforge.net/
- Sortable columns and result paging
- displaytag vs. displaytag-el

<display:table>

- name
- requestURI
- pagesize
- uid
- class

<display:column>

- property
- sortable
- title
- sortProperty

Diaplaytag Properties

- http://displaytag.sourceforge.net/11/configuration.html
- displaytag.properties for the whole application
- <display:setProperty> for a particular table