

东软睿道内部公开

文件编号: D000-

SpringMVC框架技术

版本: 3.6.0

第6章 SSM框架整合

东软睿道教育信息技术有限公司
(版权所有, 翻版必究)

Copyright © Neusoft Educational Information Technology Co., Ltd
All Rights Reserved



本章教学目标

- ✓ 了解框架整合的环境搭建；
- ✓ 了解逆向生成po类及mapper；
- ✓ 理解ssm系统架构；
- ✓ 理解框架整合思路；
- ✓ 掌握dao、service、springmvc的整合；



本章教学内容

节	知识点	掌握程度	难易程度	教学形式	对应在线微课
框架整合概述	框架整合思路	理解		线下	框架整合思路
	环境搭建	了解		线下	环境搭建
整合dao	数据库及mybatis配置文件	掌握		线下	数据库及mybatis配置文件
	applicationContext-dao.xml	掌握		线下	applicationContext-dao配置文件
	po类及mapper	了解		线下	po类及mapper
整合service	定义service	掌握		线下	定义service
	applicationContext-transaction.xml	掌握	难	线下	applicationContext-transaction配置文件
整合springmvc	springmvc.xml和web.xml	掌握		线下	springmvc和web配置文件
	编写Handler	掌握		线下	编写Handler
	编写jsp文件	掌握		线下	编写jsp文件
本章实战项目任务实现	实战项目任务实现	掌握		线下	实战项目任务实现

本章实战项目任务

- ❖ 通过本章内容的学习，使用SSM框架技术完成《跨境电商系统》完整的用户登录功能
 - ▶ 用户在登录页面输入用户名、密码，点击登录按钮
 - ▶ springmvc处理请求，调用dao验证登录信息，返回处理结果
- ❖ 运行效果如图：

用户登录

用户名:

密码:

登录

用户登录

用户名或密码错误

用户名:

密码:

登录

登录成功！

展示系统首页信息...

CONTENTS

目录

01

框架整合概述

02

整合dao

03

整合service

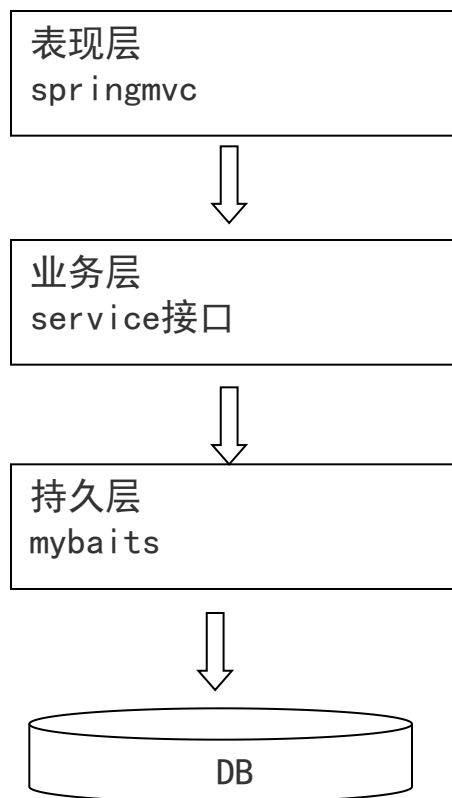
04

整合springmvc

05

本章实战项目任务实现

SpringMVC+Mybatis的系统架构



- ❖ spring将各层进行整合
 - ▶ 通过spring管理持久层的mapper (相当于dao接口)
 - ▶ 通过spring管理业务层service, service中可以调用mapper接口。
 - ▶ spring进行事务控制。
 - ▶ 通过spring管理表现层Handler, Handler中可以调用service接口。
- ❖ mapper、service、Handler都是javabean。

框架整合思路

- ❖ 第一步：整合dao层
 - ▶ mybatis和spring整合，通过spring管理mapper接口。
 - ▶ 使用mapper的扫描器自动扫描mapper接口在spring容器中进行注册。
- ❖ 第二步：整合service层
 - ▶ 通过spring管理 service接口。
 - ▶ 使用组件自动扫描将service接口纳入spring容器中管理。
 - ▶ 实现事务控制。
- ❖ 第三步：整合springmvc
 - ▶ 由于springmvc是spring的模块，不需要整合。



环境搭建

- ❖ java环境：
 - ▶ jdk1.8
 - ▶ Spring Tool Suite™ (STS)
- ❖ 数据库环境：
 - ▶ Oracle10g或者Mysql5
- ❖ 外加的jar包
 - ▶ 数据库驱动包：
 - ▶ mybatis的jar包
 - ▶ mybatis和spring整合包
 - ▶ log4j包
 - ▶ dbcp数据库连接池包
 - ▶ spring4所有jar包
 - ▶ jstl包

环境搭建

- ❖ 新建maven web工程，导入jar包
 - ▶ 导入Spring、MyBatis必须的jar包
 - ▶ 导入Spring整合MyBatis的jar包
 - ▶ 导入数据库驱动jar包

```
<dependency>
  <groupId>org.mybatis</groupId>
  <artifactId>mybatis</artifactId>
  <version>3.4.2</version>
</dependency>
```

```
<dependency>
  <groupId>org.mybatis</groupId>
  <artifactId>mybatis-spring</artifactId>
  <version>1.3.0</version>
</dependency>
```

```
<!-- -oracle 驱动 -->
<dependency>
  <groupId>com.oracle</groupId>
  <artifactId>ojdbc14</artifactId>
  <version>10.2.0.1.0</version>
</dependency>
```

- ▶ 全部代码参见：ch06-ssm01工程

课堂练习（2分钟）

- ❖ 描述springmvc+mybatis的系统架构
- ❖ 简述ssm整合思路

CONTENTS

目录

01

框架整合概述

02

整合dao

03

整合service

04

整合springmvc

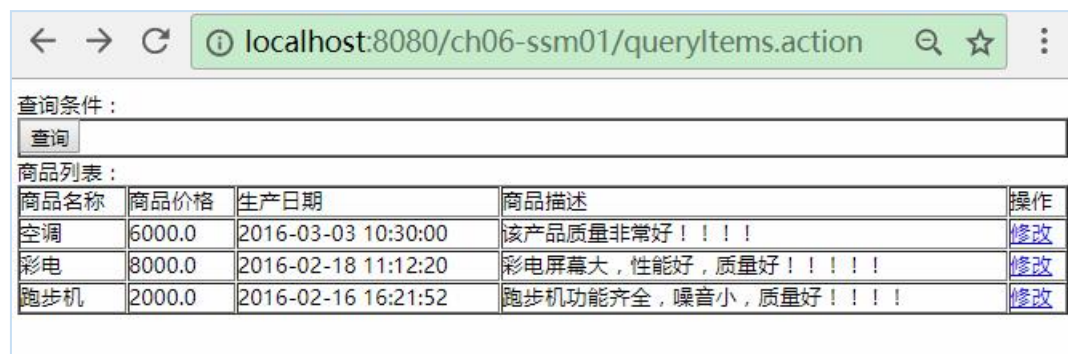
05

本章实战项目任务实现

SSM框架整合

❖ 示例：

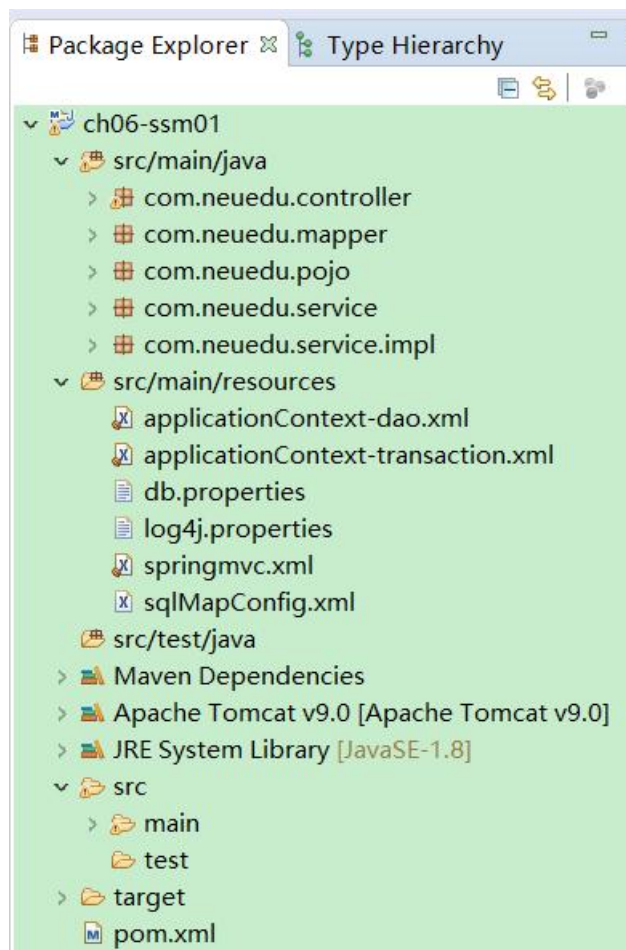
- ▶ 使用springmvc和mybatis完成商品列表查询



查询条件：				
<input type="text" value="查询"/>				
商品列表：				
商品名称	商品价格	生产日期	商品描述	操作
空调	6000.0	2016-03-03 10:30:00	该产品质量非常好！！！！	修改
彩电	8000.0	2016-02-18 11:12:20	彩电屏幕大，性能好，质量好！！！！！！	修改
跑步机	2000.0	2016-02-16 16:21:52	跑步机功能齐全，噪音小，质量好！！！！	修改

SSM框架整合

❖ 工程结构



数据库服务器配置文件

❖ 在resources下创建db.properties

```
1 jdbc.driver=oracle.jdbc.driver.OracleDriver
2 jdbc.url=jdbc:oracle:thin:@localhost:1521:orcl
3 jdbc.username=scott
4 jdbc.password=tiger
5
```

▶ 全部代码参见：ch06-ssm01工程

日志配置文件

- ❖ 在resources下创建log4j.properties

```
1 # Global logging configuration\uff0c\u5efa\u8bae\u5f00\u53d1\u73af\u5883\u4e2d\u8981\u7528debug
2 log4j.rootLogger=DEBUG, stdout
3 # Console output...
4 log4j.appender.stdout=org.apache.log4j.ConsoleAppender
5 log4j.appender.stdout.layout=org.apache.log4j.PatternLayout
6 log4j.appender.stdout.layout.ConversionPattern=%5p [%t] - %m%n
```

- ▶ 全部代码参见：ch06-ssm01工程

Mybatis配置文件

- ❖ 在resources下创建*SqlMapConfig.xml*

```
<!-- 配置别名 -->
<typeAliases>
  <!-- 批量扫描别名 -->
  <package name="com.neuedu.pojo"/>
</typeAliases>
```

- ▶ 全部代码参见：ch06-ssm01工程

applicationContext-dao.xml

- ❖ 在resources下创建applicationContext-dao.xml
 - ▶ 配置数据源

```
<!-- 加载db.properties文件中的内容，db.properties文件中key命名要有一定的特殊规则 -->
<context:property-placeholder location="classpath:db.properties" />
<!-- 配置数据源，dbcp -->

<bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource"
    destroy-method="close">
    <property name="driverClassName" value="${jdbc.driver}" />
    <property name="url" value="${jdbc.url}" />
    <property name="username" value="${jdbc.username}" />
    <property name="password" value="${jdbc.password}" />
    <property name="maxActive" value="30" />
    <property name="maxIdle" value="5" />
</bean>
```

applicationContext-dao.xml

- ❖ spring管理SqlSessionFactory、mapper
 - ▶ 配置SqlSessionFactory、mapper扫描器

```
<!-- sqlSessionFactory -->
<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
    <!-- 数据库连接池 -->
    <property name="dataSource" ref="dataSource" />
    <!-- 加载mybatis的全局配置文件 -->
    <property name="configLocation" value="classpath:sqlMapConfig.xml" />
</bean>
<!-- mapper扫描器 -->
<bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">
    <!-- 扫描包路径，如果需要扫描多个包，中间使用半角逗号隔开 -->
    <property name="basePackage" value="com.neuedu.mapper"></property>
    <property name="sqlSessionFactoryBeanName" value="sqlSessionFactory" />
</bean>
```

po类及mapper

- ❖ 逆向工程生成po类及mapper
 - ▶ 将生成的文件拷贝至工程中

- ▶ 全部代码参见：ch06-ssm01工程

```
▼ com.neuedu.mapper
  > ItemsMapper.java
  > ItemsMapperCustom.java
  > OrderdetailMapper.java
  > OrdersMapper.java
  > UserMapper.java
  > ItemsMapper.xml
  > ItemsMapperCustom.xml
  > OrderdetailMapper.xml
  > OrdersMapper.xml
  > UserMapper.xml
▼ com.neuedu.pojo
  > Items.java
  > ItemsCustom.java
  > ItemsExample.java
  > ItemsQueryVo.java
  > Orderdetail.java
  > OrderdetailExample.java
  > Orders.java
  > OrdersExample.java
  > User.java
  > UserExample.java
```

po类及mapper

❖ 手动定义商品查询mapper

- ▶ ItemsMapperCustom.xml
- ▶ ItemsMapperCustom.java

```
<!-- 定义商品查询的sql片段，就是商品查询条件 -->
<sql id="query_items_where">
  <!-- 使用动态sql，通过if判断，满足条件进行sql拼接 -->
  <!-- 商品查询条件通过ItemsQueryVo包装对象 中itemsCustom属性传递 -->
  <if test="itemsCustom!=null">
    <if test="itemsCustom.name!=null and itemsCustom.name!=''">
      items.name LIKE '%${itemsCustom.name}%'
    </if>
  </if>
</sql>

<!-- 商品列表查询 -->
<!-- parameterType传入包装对象(包装了查询条件)
      resultType建议使用扩展对象
-->
<select id="findItemsList" parameterType="com.neuedu.pojo.ItemsQueryVo"
        resultType="com.neuedu.pojo.ItemsCustom">
  SELECT items.* FROM items

  <where>
    <include refid="query_items_where"></include>
  </where>
</select>
```

po类及mapper

- ❖ ItemsMapperCustom.java
 - ▶ 定义商品信息扩展类ItemsCustom.java
 - ▶ 定义商品信息包装类ItemsQueryVo.java

```
public interface ItemsMapperCustom {  
    //商品查询列表  
    public List<ItemsCustom> findItemsList(ItemsQueryVo itemsQueryVo) throws Exception;  
}
```

- ▶ 全部代码参见：ch06-ssm01工程

课堂练习（2分钟）

- ❖ 整合dao需要配置哪些配置文件？
- ❖ 描述一下mapper扫描器的配置项

CONTENTS

目录

01

框架整合概述

02

整合dao

03

整合service

04

整合springmvc

05

本章实战项目任务实现

定义service

- ❖ Spring管理service bean
 - ▶ 通过组件自动扫描，将service交给spring容器管理

```
public interface ItemsService {  
    public List<ItemsCustom> findItemsList(ItemsQueryVo itemsQueryVo) throws Exception;  
}
```

```
@Service("itemsService")  
public class ItemsServiceImpl implements ItemsService{
```

- ❖ 全部代码参见：ch06-ssm01工程

applicationContext-transaction.xml

❖ Spring进行事务控制

- ▶ 在applicationContext-transaction.xml中使用spring声明式事务控制方法

```
<bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
    <!-- 数据源
    dataSource在applicationContext-dao.xml中配置了
    -->
    <property name="dataSource" ref="dataSource"/>
</bean>

<!-- 通知 -->
<tx:advice id="txAdvice" transaction-manager="transactionManager">
    <tx:attributes>
        <!-- 传播行为 -->
        <tx:method name="save*" propagation="REQUIRED"/>
        <tx:method name="delete*" propagation="REQUIRED"/>
        <tx:method name="insert*" propagation="REQUIRED"/>
        <tx:method name="update*" propagation="REQUIRED"/>
        <tx:method name="find*" propagation="SUPPORTS" read-only="true"/>
        <tx:method name="get*" propagation="SUPPORTS" read-only="true"/>
        <tx:method name="select*" propagation="SUPPORTS" read-only="true"/>
    </tx:attributes>
</tx:advice>
```

❖ 全部代码参见：ch06-ssm01工程

课堂练习（2分钟）

- ❖ 如何将service交给spring容器管理？
- ❖ 如何配置声明式事务？



CONTENTS

目录

01

框架整合概述

02

整合dao

03

整合service

04

整合springmvc

05

本章实战项目任务实现

springmvc.xml 配置文件

- ❖ 在resources下创建springmvc.xml
 - ▶ 配置处理器映射器、适配器、视图解析器

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

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

```
<bean  
    class="org.springframework.web.servlet.view.InternalResourceViewResolver">  
    <!-- 配置jsp路径的前缀 -->  
    <property name="prefix" value="/WEB-INF/jsp/" />  
    <!-- 配置jsp路径的后缀 -->  
    <property name="suffix" value=".jsp" />  
</bean>
```

- ▶ 全部代码参见：ch06-ssm01工程

web.xml 配置文件

❖ 配置前端控制器

```
<servlet>
  <servlet-name>springmvc</servlet-name>
  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
  <init-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath:springmvc.xml</param-value>
  </init-param>
</servlet>

<servlet-mapping>
  <servlet-name>springmvc</servlet-name>
  <url-pattern>*.action</url-pattern>
</servlet-mapping>
```

▶ 全部代码参见：ch06-ssm01工程

web.xml 配置文件

❖ 加载spring容器

- ▶ 在web.xml中，添加spring容器监听器，加载spring容器。

```
<!-- 加载spring容器 -->  
<context-param>  
    <param-name>contextConfigLocation</param-name>  
    <param-value>classpath:applicationContext-*.xml</param-value>  
</context-param>
```

```
<listener>  
    <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>  
</listener>
```

- ▶ 全部代码参见：ch06-ssm01工程

编写Handler

❖ Spring管理handler

- ▶ 通过组件自动扫描，将handler交给spring容器管理

```
@Controller
public class ItemsController {
    @Autowired
    ItemsService itemsService;

    @RequestMapping("/queryItems")
    public ModelAndView queryItems() throws Exception{
        //调用service查找 数据库，查询商品列表
        List<ItemsCustom> itemList = new ArrayList<ItemsCustom>();
        itemList = itemsService.findItemsList(null);

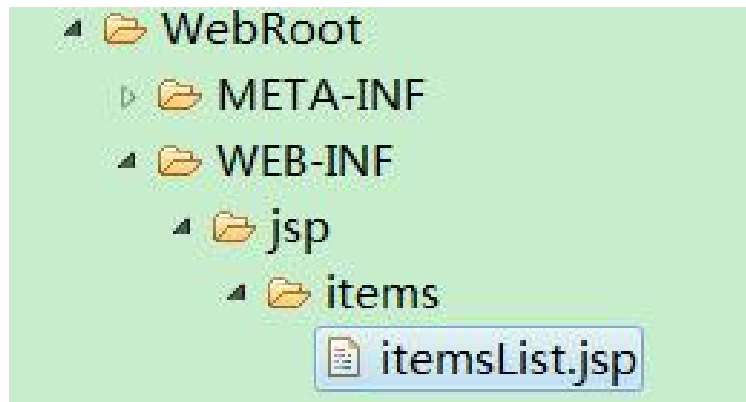
        //返回ModelAndView
        ModelAndView modelAndView = new ModelAndView();
        //相当于request的setAttribute，在jsp页面中通过itemList取数据
        modelAndView.addObject("itemList", itemList);

        modelAndView.setViewName("items/itemList");

        return modelAndView;
    }
}
```

❖ 全部代码参见：ch06-ssm01工程

编写jsp文件



❖ 全部代码参见：ch06-ssm01工程

部署测试

❖ 部署测试

<div>← → ↻ ⓘ localhost:8080/ch06-ssm01/queryItems.action 🔍 ☆ ⋮</div>				
查询条件：				
<div>查询</div>				
商品列表：				
商品名称	商品价格	生产日期	商品描述	操作
空调	6000.0	2016-03-03 10:30:00	该产品质量非常好！！！！	修改
彩电	8000.0	2016-02-18 11:12:20	彩电屏幕大，性能好，质量好！！！！	修改
跑步机	2000.0	2016-02-16 16:21:52	跑步机功能齐全，噪音小，质量好！！！！	修改

课堂练习（50分钟）

❖ 理解教学案例并独立完成



CONTENTS

目录

01

框架整合概述

02

整合dao

03

整合service

04

整合springmvc

05

本章实战项目任务实现

本章实战项目任务实现

- ❖ 通过本章内容的学习，使用SSM框架技术完成《跨境电商系统》完整的用户登录功能
 - ▶ 用户在登录页面输入用户名、密码，点击登录按钮
 - ▶ springmvc处理请求，调用dao验证登录信息，返回处理结果

❖ 运行效果如图：

用户登录

用户名:

密码:

登录

登录成功！

展示系统首页信息...

用户登录

用户名或密码错误

用户名:

密码:

登录

本章重点总结

- ❖ 了解框架整合的环境搭建；
- ❖ 了解逆向生成po类及mapper；
- ❖ 理解ssm系统架构；
- ❖ 掌握框架整合思路；
- ❖ 掌握dao、service、springmvc的整合；



课后作业【必做任务】

- ❖ 使用SSM框架技术完成《跨境电商系统》用户注册功能
 - ▶ 填写用户注册信息，点击注册按钮
 - ▶ 成功到达用户登录页面
- ❖ 运行效果如图：

用户注册

用户名:

密码:

昵称:

用户登录

用户名:

密码:

课后作业【选做任务】

- ❖ 使用SSM框架技术完成商品信息的修改
 - ▶ 在查询结果列表页面中，点击修改链接（图1）
 - ▶ 进入到修改页面，进行信息修改（图2）
 - ▶ 点击提交按钮，信息校验，有错，显示错误信息（图2）
 - ▶ 修改成功，显示最新商品列表（图3）
- ❖ 运行效果如图：

商品列表：

商品名称	商品价格	生产日期	商品描述	操作
空调	6000.0	2016-03-03 10:30:00	该产品质量非常好！！！！	修改
彩电	8000.0	2016-02-18 11:12:20	彩电屏幕大，性能好，质量好！！！！	修改
跑步机	2000.0	2016-02-16 16:21:52	跑步机功能齐全，噪音小，质量好！！！！	修改

图1

请输入1到30个字符的商品信息

修改商品信息：

商品名称	
商品价格	18888
商品生产日期	2016-02-18 11:12:20
商品简介	彩电屏幕大，性能好，质量好！！！！
提交	

图2

商品列表：

商品名称	商品价格	生产日期	商品描述	操作
空调	6000.0	2016-03-03 10:30:00	该产品质量非常好！！！！	修改
iphone8手机	8888.0	2016-02-18 11:12:20	屏幕大，性能好，质量好！！！！	修改
跑步机	2000.0	2016-02-16 16:21:52	跑步机功能齐全，噪音小，质量好！！！！	修改

图3

课后作业【线上任务】

❖ 线上任务

- ▶ 安排学员线上学习任务（安排学员到睿道实训平台进行复习和预习的任务，主要是进行微课的学习）

