

淘淘商城

第二天

**讲师：传智.入云龙**

# 课程计划

1. SSM框架整合
2. mybatis逆向工程
3. 商品列表
4. 商品列表分页处理

# SSM框架整合

## 后台系统所用的技术

框架：Spring + SpringMVC + Mybatis

前端：EasyUI

数据库：mysql

## 创建数据库

1、安装mysql数据库

2、在mysql中创建一个taotao数据库

3、导入数据库脚本。

## Mybatis逆向工程

**执行逆向工程**

使用官方网站的mapper自动生成工具mybatis-generator-core-1.3.2来生成po类和mapper映射文件。

## 整合思路

1. Dao层：

mybatis整合spring，通过spring管理SqlSessionFactory、mapper代理对象。需要mybatis和spring的整合包。

|  |  |
| --- | --- |
| 整合内容 | 对应工程 |
| Pojo | Taotao-mangaer-pojo |
| Mapper映射文件 | Taotao-mangaer-mapper |
| Mapper接口 | Taotao-mangaer-mapper |
| sqlmapConfig.xml | Taotao-manager-web |
| applicationContext-dao.xml | Taotao-manager-web |

1. Service层：

所有的实现类都放到spring容器中管理。由spring创建数据库连接池，并有spring管理实务。

|  |  |
| --- | --- |
| 整合内容 | 对应工程 |
| Service接口及实现类 | Taotao-mangaer-service |
| applicationContext-service.xml | Taotao-manager-web |
| applicationContext-trans.xml | Taotao-manager-web |

1. 表现层：

Springmvc整合spring框架，由springmvc管理controller。

|  |  |
| --- | --- |
| 整合内容 | 对应工程 |
| springmvc.xml | Taotao-manager-web |
| Controller | Taotao-manager-web |

## Dao整合

### 创建SqlMapConfig.xml配置文件

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"* ?>  <!DOCTYPE configuration  PUBLIC "-//mybatis.org//DTD Config 3.0//EN"  "http://mybatis.org/dtd/mybatis-3-config.dtd">  <configuration>  </configuration> |

### Spring整合mybatis

创建applicationContext-dao.xml

|  |
| --- |
| <beans xmlns=*"http://www.springframework.org/schema/beans"*  xmlns:context=*"http://www.springframework.org/schema/context"* xmlns:p=*"http://www.springframework.org/schema/p"*  xmlns:aop=*"http://www.springframework.org/schema/aop"* xmlns:tx=*"http://www.springframework.org/schema/tx"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.0.xsd*  *http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.0.xsd*  *http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.0.xsd http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.0.xsd*  *http://www.springframework.org/schema/util http://www.springframework.org/schema/util/spring-util-4.0.xsd"*>  <!-- 数据库连接池 -->  <!-- 加载配置文件 -->  <context:property-placeholder location=*"classpath:properties/\*.properties"* />  <!-- 数据库连接池 -->  <bean id=*"dataSource"* class=*"com.alibaba.druid.pool.DruidDataSource"*  destroy-method=*"close"*>  <property name=*"url"* value=*"${jdbc.url}"* />  <property name=*"username"* value=*"${jdbc.username}"* />  <property name=*"password"* value=*"${jdbc.password}"* />  <property name=*"driverClassName"* value=*"${jdbc.driver}"* />  <property name=*"maxActive"* value=*"10"* />  <property name=*"minIdle"* value=*"5"* />  </bean>  <!-- 让spring管理sqlsessionfactory 使用mybatis和spring整合包中的 -->  <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:mybatis/SqlMapConfig.xml"* />  </bean>  <bean class=*"org.mybatis.spring.mapper.MapperScannerConfigurer"*>  <property name=*"basePackage"* value=*"com.taotao.mapper"* />  </bean>  </beans> |

**db.properties**

|  |
| --- |
| jdbc.driver=com.mysql.jdbc.Driver  jdbc.url=jdbc:mysql://localhost:3306/taotao?characterEncoding=utf-8  jdbc.username=root  jdbc.password=root |

备注：

Druid是目前最好的数据库连接池，在功能、性能、扩展性方面，都超过其他数据库连接池，包括DBCP、C3P0、BoneCP、Proxool、JBoss DataSource。

Druid已经在阿里巴巴部署了超过600个应用，经过多年多生产环境大规模部署的严苛考验。

## Service整合

### 管理Service实现类

|  |
| --- |
| <beans xmlns=*"http://www.springframework.org/schema/beans"*  xmlns:context=*"http://www.springframework.org/schema/context"* xmlns:p=*"http://www.springframework.org/schema/p"*  xmlns:aop=*"http://www.springframework.org/schema/aop"* xmlns:tx=*"http://www.springframework.org/schema/tx"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.0.xsd*  *http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.0.xsd*  *http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.0.xsd http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.0.xsd*  *http://www.springframework.org/schema/util http://www.springframework.org/schema/util/spring-util-4.0.xsd"*>  <context:component-scan base-package=*"com.taotao.service"*/>  </beans> |

### 事务管理

创建applicationContext-trans.xml

|  |
| --- |
| <beans xmlns=*"http://www.springframework.org/schema/beans"*  xmlns:context=*"http://www.springframework.org/schema/context"* xmlns:p=*"http://www.springframework.org/schema/p"*  xmlns:aop=*"http://www.springframework.org/schema/aop"* xmlns:tx=*"http://www.springframework.org/schema/tx"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.0.xsd*  *http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.0.xsd*  *http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.0.xsd http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-4.0.xsd*  *http://www.springframework.org/schema/util http://www.springframework.org/schema/util/spring-util-4.0.xsd"*>  <!-- 事务管理器 -->  <bean id=*"transactionManager"*  class=*"org.springframework.jdbc.datasource.DataSourceTransactionManager"*>  <!-- 数据源 -->  <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=*"insert\*"* propagation=*"REQUIRED"* />  <tx:method name=*"add\*"* propagation=*"REQUIRED"* />  <tx:method name=*"create\*"* propagation=*"REQUIRED"* />  <tx:method name=*"delete\*"* propagation=*"REQUIRED"* />  <tx:method name=*"update\*"* propagation=*"REQUIRED"* />  <tx:method name=*"find\*"* propagation=*"SUPPORTS"* read-only=*"true"* />  <tx:method name=*"select\*"* propagation=*"SUPPORTS"* read-only=*"true"* />  <tx:method name=*"get\*"* propagation=*"SUPPORTS"* read-only=*"true"* />  </tx:attributes>  </tx:advice>  <!-- 切面 -->  <aop:config>  <aop:advisor advice-ref=*"txAdvice"*  pointcut=*"execution(\* com.taotao.service.\*.\*(..))"* />  </aop:config>  </beans> |

## 表现层整合

### Springmvc.xml

|  |
| --- |
| <?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:p=*"http://www.springframework.org/schema/p"*  xmlns:context=*"http://www.springframework.org/schema/context"*  xmlns:mvc=*"http://www.springframework.org/schema/mvc"*  xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd*  *http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc-4.0.xsd*  *http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd"*>  <context:component-scan base-package=*"com.taotao.controller"* />  <mvc:annotation-driven />  <bean  class=*"org.springframework.web.servlet.view.InternalResourceViewResolver"*>  <property name=*"prefix"* value=*"/WEB-INF/jsp/"* />  <property name=*"suffix"* value=*".jsp"* />  </bean>  </beans> |

### web.xml

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*  xmlns=*"http://java.sun.com/xml/ns/javaee"* xmlns:web=*"http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"*  xsi:schemaLocation=*"http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"*  id=*"WebApp\_ID"* version=*"2.5"*>  <display-name>taotao-manager-web</display-name>  <welcome-file-list>  <welcome-file>login.html</welcome-file>  </welcome-file-list>  <!-- 加载spring容器 -->  <context-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:spring/applicationContext\*.xml</param-value>  </context-param>  <listener>  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>  </listener>  <!-- 解决post乱码 -->  <filter>  <filter-name>CharacterEncodingFilter</filter-name>  <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>  <init-param>  <param-name>encoding</param-name>  <param-value>utf-8</param-value>  </init-param>  <!-- <init-param>  <param-name>forceEncoding</param-name>  <param-value>true</param-value>  </init-param> -->  </filter>  <filter-mapping>  <filter-name>CharacterEncodingFilter</filter-name>  <url-pattern>/\*</url-pattern>  </filter-mapping>  <!-- springmvc的前端控制器 -->  <servlet>  <servlet-name>taotao-manager</servlet-name>  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  <!-- contextConfigLocation不是必须的， 如果不配置contextConfigLocation， springmvc的配置文件默认在：WEB-INF/servlet的name+"-servlet.xml" -->  <init-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:spring/springmvc.xml</param-value>  </init-param>  <load-on-startup>1</load-on-startup>  </servlet>  <servlet-mapping>  <servlet-name>taotao-manager</servlet-name>  <url-pattern>/</url-pattern>  </servlet-mapping>  </web-app> |

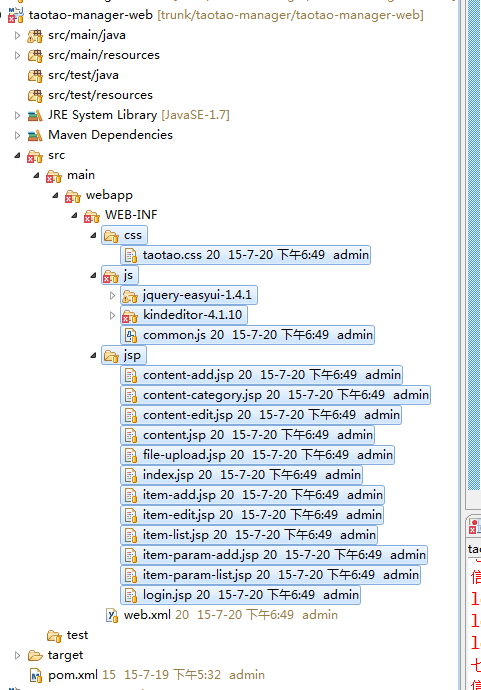
### 整合静态页面

静态页面位置：02.第二天（三大框架整合，后台系统搭建）\01.参考资料\后台管理系统静态页面



**使用方法:**

把静态页面添加到taotao-manager-web工程中的WEB-INF下：



由于在web.xml中定义的url拦截形式为“/”表示拦截所有的url请求，包括静态资源例如css、js等。所以需要在springmvc.xml中添加资源映射标签：

|  |
| --- |
| <mvc:resources location=*"/WEB-INF/js/"* mapping=*"/js/\*\*"*/>  <mvc:resources location=*"/WEB-INF/css/"* mapping=*"/css/\*\*"*/> |

## 修改taotao-manager-mapper的pom文件

在pom文件中添加如下内容：

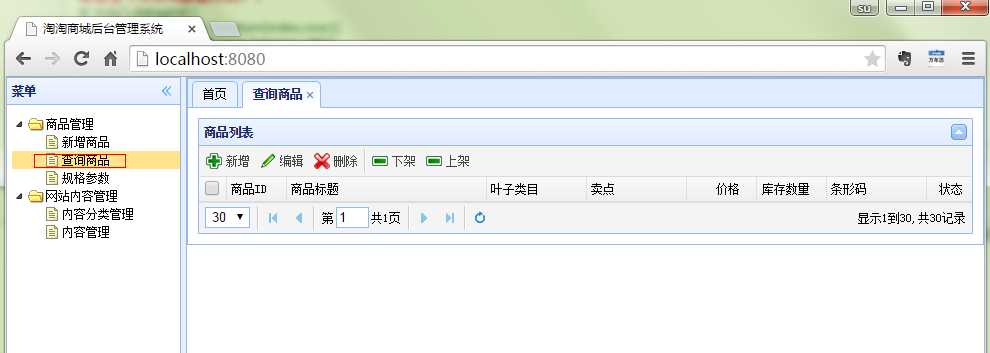
|  |
| --- |
| <!-- 如果不添加此节点mybatis的mapper.xml文件都会被漏掉。 -->  <build>  <resources>  <resource>  <directory>src/main/java</directory>  <includes>  <include>\*\*/\*.properties</include>  <include>\*\*/\*.xml</include>  </includes>  <filtering>false</filtering>  </resource>  </resources>  </build> |

## 整合测试

根据商品id查询商品信息。

# 商品列表查询

## 商品列表页面



**对应的jsp为**：

item-list.jsp

**请求的url：**

/item/list

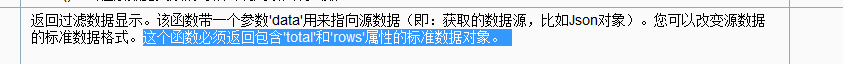
**请求的参数：**

page=1&rows=30

**响应的json数据格式：**

Easyui中datagrid控件要求的数据格式为：

{total:”2”,rows:[{“id”:”1”,”name”,”张三”},{“id”:”2”,”name”,”李四”}]}



## **响应的json数据格式**EasyUIResult

|  |
| --- |
| **public** **class** EasyUIResult {  **private** Integer total;  **private** List<?> rows;  **public** EasyUIResult(Integer total, List<?> rows) {  **this**.total = total;  **this**.rows = rows;  }  **public** EasyUIResult(Long total, List<?> rows) {  **this**.total = total.intValue();  **this**.rows = rows;  }  **public** Integer getTotal() {  **return** total;  }  **public** **void** setTotal(Integer total) {  **this**.total = total;  }  **public** List<?> getRows() {  **return** rows;  }  **public** **void** setRows(List<?> rows) {  **this**.rows = rows;  }  } |

## 分页处理

### Mybatis分页插件 - PageHelper说明

如果你也在用Mybatis，建议尝试该分页插件，这个一定是最方便使用的分页插件。

该插件目前支持Oracle,Mysql,MariaDB,SQLite,Hsqldb,PostgreSQL六种数据库分页。

### 使用方法

第一步：在Mybatis配置xml中配置拦截器插件:

<plugins>

<!-- com.github.pagehelper为PageHelper类所在包名 -->

<plugin interceptor="com.github.pagehelper.PageHelper">

<!-- 设置数据库类型 Oracle,Mysql,MariaDB,SQLite,Hsqldb,PostgreSQL六种数据库-->

<property name="dialect" value="mysql"/>

</plugin>

</plugins>

第二步：在代码中使用

1、设置分页信息：

//获取第1页，10条内容，默认查询总数count

PageHelper.startPage(1, 10);

//紧跟着的第一个select方法会被分页

List<Country> list = countryMapper.selectIf(1);

2、取分页信息

//分页后，实际返回的结果list类型是Page<E>，如果想取出分页信息，需要强制转换为Page<E>，

Page<Country> listCountry = (Page<Country>)list;

listCountry.getTotal();

1. 取分页信息的第二种方法

//获取第1页，10条内容，默认查询总数count

PageHelper.startPage(1, 10);

List<Country> list = countryMapper.selectAll();

//用PageInfo对结果进行包装

PageInfo page = new PageInfo(list);

//测试PageInfo全部属性

//PageInfo包含了非常全面的分页属性

assertEquals(1, page.getPageNum());

assertEquals(10, page.getPageSize());

assertEquals(1, page.getStartRow());

assertEquals(10, page.getEndRow());

assertEquals(183, page.getTotal());

assertEquals(19, page.getPages());

assertEquals(1, page.getFirstPage());

assertEquals(8, page.getLastPage());

assertEquals(true, page.isFirstPage());

assertEquals(false, page.isLastPage());

assertEquals(false, page.isHasPreviousPage());

assertEquals(true, page.isHasNextPage());

## Mapper

使用逆向工程生成的mapper文件。

## Service

|  |
| --- |
| @Service  **public** **class** ItemServiceImpl **implements** ItemService {  @Autowired  **private** TbItemMapper itemMapper;  @Override  **public** EasyUIResult getItemList(Integer page, Integer rows) **throws** Exception {  TbItemExample example = **new** TbItemExample();  //设置分页  PageHelper.*startPage*(page, rows);  List<TbItem> list = itemMapper.selectByExample(example);  //取分页信息  PageInfo<TbItem> pageInfo = **new** PageInfo<>(list);  **long** total = pageInfo.getTotal();  EasyUIResult result = **new** EasyUIResult(total, list);    **return** result;  }  } |

## Controller

|  |
| --- |
| @Controller  @RequestMapping("/item")  **public** **class** ItemController {    @Autowired  **private** ItemService itemService;  @RequestMapping("/list")  //设置相应的内容为json数据  @ResponseBody  **public** EasyUIResult getItemlist(@RequestParam(defaultValue="1")Integer page,  @RequestParam(defaultValue="30")Integer rows) **throws** Exception {  //查询商品列表  EasyUIResult result = itemService.getItemList(page, rows);    **return** result;  }  } |

