武汉纺织大学

Web应用开发课程设计

**学 院： 数学与计算机学院**

**班 级： 软件工程11804**

**姓 名： 许志懿**

**学 号： 1802230219**

**指导老师： 聂刚**

**成 绩：**

**完成日期： 2020年12月15日**

**1 需求分析**

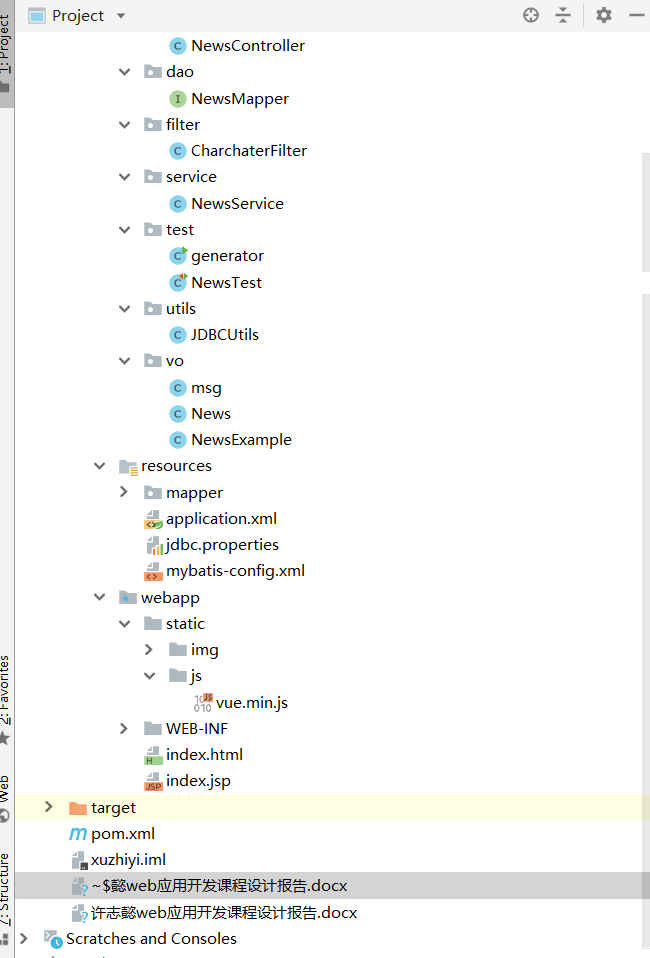
按图设计实现新闻显示页面，页面所需图片，文字均来自数据库。



# **2 系统设计**

# **3 系统实现**

## 3.1 项目结构



## 3.2 配置文件

在项目下新建Source Folder，名为resources

### 3.2.1 jdbc.properties文件

该配置文件主要是以文件形式保存数据库的驱动类名称，连接数据库的URL地址，访问数据库的用户名及对应的密码，程序运行时会读取该文件相关信息，避免硬编码，当相关信息发生变化时，只需修改配置文件而不用修改源代码，增加程序的可扩展型。

**jdbc.username**=**root  
jdbc.password**=**root  
jdbc.url**=**jdbc:mysql://47.110.70.107:8081/news?serverTimezone=UTC&useUnicode=true&characterEncoding=utf8  
jdbc.className**=**com.mysql.jdbc.Driver**

### 3.2.2 application.xml文件

进行数据库配置连接池，配置mybatis 插件和扫描mapper映射文件，然后进行事务的配置

*<?***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:context="http://www.springframework.org/schema/context"  
 xmlns:aop="http://www.springframework.org/schema/aop"  
 xmlns:tx="http://www.springframework.org/schema/tx"  
 xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd  
http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd  
http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop.xsd  
http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx.xsd"**>  
 <**context:component-scan base-package="com.xuzhiyi"**>  
 <**context:exclude-filter type="annotation" expression="org.springframework.stereotype.Controller"**/>  
 </**context:component-scan**>  
 <**context:property-placeholder location="classpath:jdbc.properties"**/>  
 *<!-- 配置连接池-->* <**bean class="com.mchange.v2.c3p0.ComboPooledDataSource" id="pooledDataSource"**>  
 <**property name="jdbcUrl" value="${jdbc.url}"**></**property**>  
 <**property name="driverClass" value="${jdbc.className}"**></**property**>  
 <**property name="user" value="${jdbc.username}"**/>  
 <**property name="password" value="${jdbc.password}"**></**property**>  
 <**property name="maxPoolSize" value="1000"**/>  
 <**property name="minPoolSize" value="3"**/>  
 *<!--关闭连接后不自动commit-->* <**property name="autoCommitOnClose" value="false"**/>  
 *<!--获取连接超时时间-->* <**property name="checkoutTimeout" value="10000"**/>  
 *<!--当获取连接失败重试次数-->* <**property name="acquireRetryAttempts" value="5"**/>  
 </**bean**>  
*<!-- 配置sqlsessionfactory-->* <**bean class="org.mybatis.spring.SqlSessionFactoryBean" id="sqlSessionFactory"**>  
 <**property name="configLocation" value="classpath:mybatis-config.xml"**/>  
 <**property name="dataSource" ref="pooledDataSource"**/>  
 <**property name="mapperLocations" value="classpath:mapper/\*.xml"**/>  
 *<!--配置mybatis 插件-->* </**bean**>  
  
  
*<!-- 扫描mapper文件-->* <**bean class="org.mybatis.spring.mapper.MapperScannerConfigurer"**>  
 <**property name="basePackage" value="com.qxx.dao"**/>  
 </**bean**>  
*<!-- 配置事务-->* <**bean class="org.springframework.jdbc.datasource.DataSourceTransactionManager" id="transactionManager"**>  
 <**property name="dataSource" ref="pooledDataSource"**/>  
 </**bean**>  
 <**aop:config**>  
 <**aop:pointcut id="txPoint" expression="execution(\* com.xuzhiyi.service..\*(..))"**/>  
 <**aop:advisor advice-ref="txAdvice" pointcut-ref="txPoint"**/>  
 </**aop:config**>  
 <**tx:advice id="txAdvice"**>  
 <**tx:attributes**>  
 <**tx:method name="\*"**/>  
 <**tx:method name="get\*"**/>  
 </**tx:attributes**>  
 </**tx:advice**>  
</**beans**>

### 3.2.3 mybatis-config.xml文件

指定资源文件，指定 MyBatis 所用日志的具体实现，未指定时将自动查找，进行JDBC的事务管理，配置mapper的映射

*<?***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**>  
 <**settings**>  
 <**setting name="mapUnderscoreToCamelCase" value="true"**/>  
 <**setting name="logImpl" value="STDOUT\_LOGGING"**/>  
 </**settings**>  
 <**typeAliases**>  
 <**package name="com.xuzhiyi.bean"**/>  
 </**typeAliases**>  
  
  
  
</**configuration**>

<?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">

<!-- 根标签 -->

<configuration>

<!-- 指定 资源文件 -->

<properties resource=*"jdbc.properties"*></properties>

<!-- 指定 MyBatis 所用日志的具体实现，未指定时将自动查找-->

<settings>

<setting name=*"logImpl"* value=*"LOG4J"* />

</settings>

<!-- 环境，可以配置多个，default：指定采用哪个环境 -->

<environments default=*"test"*>

<!-- id：唯一标识 -->

<environment id=*"test"*>

<!-- 事务管理器，JDBC类型的事务管理器 -->

<transactionManager type=*"JDBC"* />

<!-- 数据源，池类型的数据源 -->

<dataSource type=*"POOLED"*>

<property name=*"driver"* value=*"${DBDRIVER}"* />

<property name=*"url"* value=*"${DBURL}"* />

<property name=*"username"* value=*"${DBUSER}"* />

<property name=*"password"* value=*"${PASSWORD}"* />

</dataSource>

</environment>

</environments>

<!-- 配置mapper映射文件 -->

<mappers>

<mapper resource=*"dao/mapper/UserMapper.xml"* />

</mappers>

</configuration>

## 3.3 VO类

3.3.1msg.java类

设置前后端交互的状态码，看数据是否返回成功，状态码为200为返回成功，状态码为404，则返回失败。

**package** vo;  
  
**import** java.util.HashMap;  
**import** java.util.Map;  
  
*//通用返回json数据类***public class** msg {  
**private int code**;*//返回状态码***private** String **msginfo**;*//返回提示信息***private** Map<String,Object> **extend**=**new** HashMap<String,Object>();*//返回用户信息* **public static** msg success()*//数据获得成功* {  
 msg result = **new** msg();  
 result.setCode(200);  
 result.setMsginfo(**"处理成功"**);  
 **return** result;  
 }  
 **public static** msg fail()*//数据获得失败* {  
 msg result = **new** msg();  
 result.setCode(404);  
 result.setMsginfo(**"处理失败"**);  
 **return** result;  
 }  
 **public** msg add(String key,Object value)*//链式操作放入数据信息* {  
 **this**.getExtend().put(key,value);  
 **return this**;  
 }  
 **public int** getCode() {  
 **return code**;  
 }  
  
 **public void** setCode(**int** code) {  
 **this**.**code** = code;  
 }  
  
 **public** String getMsginfo() {  
 **return msginfo**;  
 }  
  
 **public void** setMsginfo(String msginfo) {  
 **this**.**msginfo** = msginfo;  
 }  
  
 **public** Map<String, Object> getExtend() {  
 **return extend**;  
 }  
  
 **public void** setExtend(Map<String, Object> extend) {  
 **this**.**extend** = extend;  
 }  
}

3.3.2News.java类

News的实体内，封装News的数据

**package** vo;  
  
**import** java.io.Serializable;  
  
**public class** News **implements** Serializable {  
 **private** Integer **id**;  
  
 **private** String **section**;  
  
 **private** String **href**;  
  
 **private** Integer **pid**;  
  
 **private** String **showType**;  
  
 **private** String **summary**;  
  
 **private** String **imgSrc**;  
  
 **private** String **artical**;  
  
 **private static final long *serialVersionUID*** = 1L;  
  
 **public** Integer getId() {  
 **return id**;  
 }  
  
 **public void** setId(Integer id) {  
 **this**.**id** = id;  
 }  
  
 **public** String getSection() {  
 **return section**;  
 }  
  
 **public void** setSection(String section) {  
 **this**.**section** = section == **null** ? **null** : section.trim();  
 }  
  
 **public** String getHref() {  
 **return href**;  
 }  
  
 **public void** setHref(String href) {  
 **this**.**href** = href == **null** ? **null** : href.trim();  
 }  
  
 **public** Integer getPid() {  
 **return pid**;  
 }  
  
 **public void** setPid(Integer pid) {  
 **this**.**pid** = pid;  
 }  
  
 **public** String getShowType() {  
 **return showType**;  
 }  
  
 **public void** setShowType(String showType) {  
 **this**.**showType** = showType == **null** ? **null** : showType.trim();  
 }  
  
 **public** String getSummary() {  
 **return summary**;  
 }  
  
 **public void** setSummary(String summary) {  
 **this**.**summary** = summary == **null** ? **null** : summary.trim();  
 }  
  
 **public** String getImgSrc() {  
 **return imgSrc**;  
 }  
  
 **public void** setImgSrc(String imgSrc) {  
 **this**.**imgSrc** = imgSrc == **null** ? **null** : imgSrc.trim();  
 }  
  
 **public** String getArtical() {  
 **return artical**;  
 }  
  
 **public void** setArtical(String artical) {  
 **this**.**artical** = artical == **null** ? **null** : artical.trim();  
 }  
}

3.4 DAO接口类NewsMapper.java

实现对数据库的增删查改

**package** dao;  
  
  
**import** java.util.List;  
  
**import** vo.News;  
**import** vo.NewsExample;  
**import** org.apache.ibatis.annotations.Param;  
  
**public interface** NewsMapper {  
 **long** countByExample(NewsExample example);  
  
 **int** deleteByExample(NewsExample example);  
  
 **int** deleteByPrimaryKey(Integer id);  
  
 **int** insert(News record);  
  
 **int** insertSelective(News record);  
  
 List<News> selectByExampleWithBLOBs(NewsExample example);  
  
 List<News> selectByExample(NewsExample example);  
  
 News selectByPrimaryKey(Integer id);  
  
 **int** updateByExampleSelective(@Param(**"record"**) News record, @Param(**"example"**) NewsExample example);  
  
 **int** updateByExampleWithBLOBs(@Param(**"record"**) News record, @Param(**"example"**) NewsExample example);  
  
 **int** updateByExample(@Param(**"record"**) News record, @Param(**"example"**) NewsExample example);  
  
 **int** updateByPrimaryKeySelective(News record);  
  
 **int** updateByPrimaryKeyWithBLOBs(News record);  
  
 **int** updateByPrimaryKey(News record);  
}

## 3.6 过滤器包Util

3.6.1 CharchaterFilter.java

解决全站乱码问题，将全站所有的编码设置为utf-8

**package** filter;  
  
**import** javax.servlet.\*;  
**import** javax.servlet.annotation.WebFilter;  
**import** javax.servlet.http.HttpServletRequest;  
**import** javax.servlet.http.HttpServletResponse;  
**import** java.io.IOException;  
  
*/\*\*  
 \* 解决全站乱码问题，处理所有的请求  
 \*/*@WebFilter(**"/\*"**)  
**public class** CharchaterFilter **implements** Filter {  
 @Override  
 **public void** init(FilterConfig filterConfig) **throws** ServletException {  
  
 }  
  
 @Override  
 **public void** doFilter(ServletRequest req, ServletResponse rep, FilterChain filterChain) **throws** IOException, ServletException {  
 *//将父接口转为子接口* HttpServletRequest request = (HttpServletRequest) req;  
 HttpServletResponse response = (HttpServletResponse) rep;  
 *//获取请求方法* String method = request.getMethod();  
 *//解决post请求中文数据乱码问题* **if**(method.equalsIgnoreCase(**"post"**)){  
 request.setCharacterEncoding(**"utf-8"**);  
 }  
 *//处理响应乱码* response.setContentType(**"text/html;charset=utf-8"**);  
 filterChain.doFilter(request,response);  
 }  
  
 @Override  
 **public void** destroy() {  
  
 }  
}

## 3.7 服务层NewsService.java

实现业务逻辑

**package** service;  
  
**import** vo.News;  
**import** vo.NewsExample;  
**import** dao.NewsMapper;  
**import** org.springframework.beans.factory.annotation.Autowired;  
**import** org.springframework.stereotype.Service;  
  
**import** java.util.HashMap;  
**import** java.util.List;  
**import** java.util.Map;  
  
@Service  
**public class** NewsService {  
 @Autowired  
 NewsMapper **newsMapper**;  
  
 **public** Map<Integer,Object> getColumns() {  
 NewsExample newsExample = **new** NewsExample();  
 NewsExample.Criteria criteria = newsExample.createCriteria();  
 criteria.andPidEqualTo(0);  
 criteria.andShowTypeEqualTo(**"0"**);  
 List<News> type0List = **newsMapper**.selectByExample(newsExample);  
 NewsExample newsExample2 = **new** NewsExample();  
 NewsExample.Criteria criteria2 = newsExample2.createCriteria();  
 criteria2.andPidEqualTo(0);  
 criteria2.andShowTypeEqualTo(**"-1"**);  
 List<News> typeSpecialList = **newsMapper**.selectByExample(newsExample2);  
 Map<Integer,Object> map = **new** HashMap<>();  
 map.put(0,type0List);  
 map.put(-1,typeSpecialList);  
 **return** map;  
 }  
  
 **public** List<News> getAll() {  
 NewsExample newsExample = **new** NewsExample();  
 **return newsMapper**.selectByExample(newsExample);  
 }  
  
 **public** Map<Integer, Object> getSectionContent(Integer id){  
 NewsExample newsExample = **new** NewsExample();  
 NewsExample.Criteria criteria = newsExample.createCriteria();  
 criteria.andPidEqualTo(id);  
 criteria.andShowTypeEqualTo(**"1"**);  
 List<News> type1List = **newsMapper**.selectByExample(newsExample);  
 NewsExample newsExample2 = **new** NewsExample();  
 NewsExample.Criteria criteria2 = newsExample2.createCriteria();  
 criteria2.andPidEqualTo(id);  
 criteria2.andShowTypeEqualTo(**"2"**);  
 List<News> type2List = **newsMapper**.selectByExample(newsExample2);  
 Map<Integer,Object> map = **new** HashMap<>();  
 map.put(1,type1List);  
 map.put(2,type2List);  
 **return** map;  
 }  
}

## 3.8 控制器NewsController

将前端返回的新闻数据返回到后端，由控制器发送到service层，service层调用dao层完成数据的增删查改。

**package** controller;  
  
  
**import** vo.msg;  
**import** org.springframework.beans.factory.annotation.Autowired;  
**import** org.springframework.web.bind.annotation.GetMapping;  
**import** org.springframework.web.bind.annotation.RestController;  
**import** service.NewsService;  
  
**import** java.util.Map;  
  
@RestController  
**public class** NewsController {  
 @Autowired  
 NewsService **newsService**;  
  
 @GetMapping(**"/getColumns.do"**)  
 **public** msg getColumns(){  
 Map<Integer, Object> columnsList = **newsService**.getColumns();  
 **return** msg.*success*().add(**"columnList"**,columnsList);  
 }  
 @GetMapping(**"/getSection.do"**)  
 **public** msg getSectionContent(Integer id){  
 Map<Integer, Object> sectionContentList = **newsService**.getSectionContent(id);  
 **return** msg.*success*().add(**"sectionContentList"**,sectionContentList);  
 }  
}

# **4 系统测试**

# **5 系统总结**