**每日作业卷**

**SpringMVC第2天**

极客营

# 关卡1

## 训练案例1

### 训练描述

使用已经提供的SpringMVC02-study工程中实现商品管理功能

要求

1. 实现商品多选，使用包装对象和参数两种方式接收商品id的数组。

2. 实现商品批量修改，提交到后台使用List接收，并在控制台打印商品信息

3. 改造搜索商品功能，实现只接收POST请求。

4. 实现上传商品图片功能

5. 使用注解的方式实现商品数据的json交互

6. 使用注解的方式接收请求url上的数据

### 操作步骤答案

1. 实现商品多选，使用包装对象和参数两种方式接收商品id的数组。

修改itemList.jsp，增加商品多选功能

<table width=*"100%"* border=*1*>

<tr>

<td>选择</td>

<td>商品名称</td>

<td>商品价格</td>

<td>生产日期</td>

<td>商品描述</td>

<td>操作</td>

</tr>

<c:forEach items=*"*${itemList }*"* var=*"item"*>

<tr>

<td><input type=*"checkbox"* name=*"ids"* value=*"*${item.id}*"*/></td>

<td>${item.name }</td>

<td>${item.price }</td>

<td><fmt:formatDate value=*"*${item.createtime}*"* pattern=*"yyyy-MM-dd HH:mm:ss"*/></td>

<td>${item.detail }</td>

编写包装对象QueryVo

**public** **class** QueryVo {

**private** Integer[] ids;

**public** Integer[] getIds() {

**return** ids;

}

**public** **void** setIds(Integer[] ids) {

**this**.ids = ids;

}

}

编写Controller

@RequestMapping("queryItem")

**public** String queryItem(QueryVo queryVo, Integer[] ids) {

System.***out***.println(queryVo.getIds().length);

System.***out***.println(ids.length);

**return** "success";

}

1. 实现商品批量修改，提交到后台使用List接收，并在控制台打印商品信息

修改itemList.jsp，增加批量修改商品功能

<c:forEach items=*"*${itemList }*"* var=*"item"* varStatus=*"s"*>

<tr>

<td><input type=*"checkbox"* name=*"ids"* value=*"*${item.id}*"*/></td>

<td>

<input type=*"hidden"* name=*"itemList[*${s.index}*].id"* value=*"*${item.id }*"*/>

<input type=*"text"* name=*"itemList[*${s.index}*].name"* value=*"*${item.name }*"*/>

</td>

<td><input type=*"text"* name=*"itemList[*${s.index}*].price"* value=*"*${item.price }*"*/></td>

<td><input type=*"text"* name=*"itemList[*${s.index}*].createtime"* value=*"*<fmt:formatDate value="${item.createtime}" pattern=*"yyyy-MM-dd HH:mm:ss"*/>*"*/></td>

<td><input type=*"text"* name=*"itemList[*${s.index}*].detail"* value=*"*${item.detail }*"*/></td>

<td><a href=*"*${pageContext.request.contextPath }*/itemEdit.action?id=*${item.id}*"*>修改</a></td>

</tr>

</c:forEach>

编写QueryVo

**public** **class** QueryVo {

**private** Integer[] ids;

**private** List<Item> itemList;

**public** List<Item> getItemList() {

**return** itemList;

}

**public** **void** setItemList(List<Item> itemList) {

**this**.itemList = itemList;

}

**public** Integer[] getIds() {

**return** ids;

}

**public** **void** setIds(Integer[] ids) {

**this**.ids = ids;

}

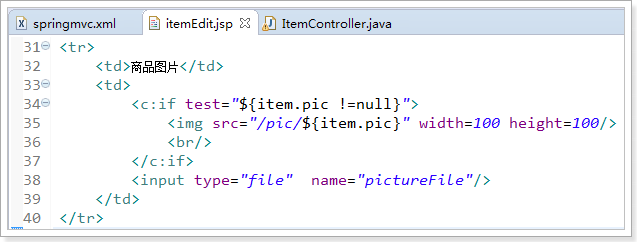
}

1. 改造搜索商品功能，实现只接收POST请求。

修改Controller方法上的注解

@RequestMapping(value = "queryItem",method = RequestMethod.***GET***)

1. 实现上传商品图片功能

配置Tomcat的虚拟路径，打开itemEdit.jsp的图片上传功能

加入上传组件的jar包

commons-fileupload和commons-io两个jar包

配置上传解析器

<!-- 文件上传,id必须设置为multipartResolver -->

<bean id=*"multipartResolver"*

class=*"org.springframework.web.multipart.commons.CommonsMultipartResolver"*>

<!-- 设置文件上传大小 -->

<property name=*"maxUploadSize"* value=*"5000000"* />

</bean>

编写上传图片Controller

@RequestMapping("updateItem")

**public** String updateItem(Item item, MultipartFile pictureFile) **throws** Exception {

// 重命名图片

UUID uuid = UUID.*randomUUID*();

String oname = pictureFile.getOriginalFilename();

String extName = oname.substring(oname.lastIndexOf("."));

String picName = uuid + extName;

pictureFile.transferTo(**new** File("C:/images/" + picName));

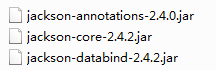
item.setPic(picName);

**this**.itemService.updateItem(item);

**return** "success";

}

1. 使用注解的方式实现商品数据的json交互

加入支持json的jar包

编写Controller方法

@RequestMapping("jsonTest")

@ResponseBody

**public** Item test(@RequestBody Item item) {

**return** item;

}

6. 使用注解的方式接收请求url上的数据

@RequestMapping("item/{id}")

**public** String queryItemById(@PathVariable("id") Integer id, Model model) {

Item item = **this**.itemService.queryItemById(id);

// 设置数据到模型中

model.addAttribute("item", item);

// 返回逻辑视图名

**return** "itemEdit";

}

# 关卡2

## 训练案例1

### 训练描述

在关卡1的商品管理基础上进行扩展加强

要求

1. 在修改商品成功后，使用返回void的方式，实现重定向和转发到商品修改页面

2. 在修改商品成功后，使用返回字符串的方式，实现转发和转发到商品修改页面

3. 实现全局异常处理器，并能够处理预期异常和运行时异常

4. 实现用户登录拦截，只有登陆成功的用户可以访问商品管理页，没有登录的用户跳转到登陆界面

### 操作步骤答案

1. 在修改商品成功后，使用返回字符串的方式，实现重定向到商品修改页面

//使用request进行转发 request.getRequestDispatcher("/itemEdit.action").forward(request, response);

// 使用response进行重定向

response.sendRedirect("/SpringMVC02-study/itemEdit.action?id="+item.getId());

1. 在修改商品成功后，使用返回字符串的方式，实现转发到商品修改页面

// 使用转发

**return** "forward:/itemEdit.action";

// 使用重定向

**return** "redirect:/itemEdit.action?id=" + item.getId();

1. 实现全局异常处理器，并能够处理预期异常和运行时异常

编写自定义异常

**public** **class** MyException **extends** Exception {

// 异常信息

**private** String message;

**public** MyException() {

**super**();

}

**public** MyException(String message) {

**super**();

**this**.message = message;

}

**public** String getMessage() {

**return** message;

}

**public** **void** setMessage(String message) {

**this**.message = message;

}

}

编写异常处理器

**public** **class** CustomHandleException **implements** HandlerExceptionResolver {

@Override

**public** ModelAndView resolveException(HttpServletRequest request, HttpServletResponse response, Object handler,

Exception exception) {

// 定义异常信息

String msg;

// 判断异常类型

**if** (exception **instanceof** MyException) {

// 如果是自定义异常，读取异常信息

msg = exception.getMessage();

} **else** {

// 如果是运行时异常，则取错误堆栈，从堆栈中获取异常信息

Writer out = **new** StringWriter();

PrintWriter s = **new** PrintWriter(out);

exception.printStackTrace(s);

msg = out.toString();

}

// 返回错误页面

ModelAndView modelAndView = **new** ModelAndView();

modelAndView.addObject("msg", msg);

modelAndView.setViewName("error");

**return** modelAndView;

}

}

4. 实现用户登录拦截，只有登陆成功的用户可以访问商品管理页，没有登录的用户跳转到登陆界面

编写Controller

@Controller

@RequestMapping("user")

**public** **class** UserController {

/\*\*

\* 跳转到登录页面

\*

\* **@return**

\*/

@RequestMapping("toLogin")

**public** String toLogin() {

**return** "login";

}

/\*\*

\* 用户登录

\*

\* **@param** username

\* **@param** password

\* **@param** session

\* **@return**

\*/

@RequestMapping("login")

**public** String login(String username, String password, HttpSession session) {

// 校验用户登录

System.***out***.println(username);

System.***out***.println(password);

// 把用户名放到session中

session.setAttribute("username", username);

**return** "redirect:/item/itemList.action";

}

}

编写拦截器

**public** **class** UserInterceptor **implements** HandlerInterceptor {

@Override

**public** **void** afterCompletion(HttpServletRequest arg0, HttpServletResponse arg1, Object arg2, Exception arg3)

**throws** Exception {

// **TODO** Auto-generated method stub

}

@Override

**public** **void** postHandle(HttpServletRequest arg0, HttpServletResponse arg1, Object arg2, ModelAndView arg3)

**throws** Exception {

}

@Override

**public** **boolean** preHandle(HttpServletRequest request, HttpServletResponse response, Object arg2) **throws** Exception {

HttpSession session = request.getSession();

String username = (String) session.getAttribute("username");

**if** (username == **null**) {

response.sendRedirect("/SpringMVC02-study/user/toLogin.action");

**return** **false**;

}

// 如果不为空则放行

**return** **true**;

}

}

配置拦截器

<mvc:interceptors>

<mvc:interceptor>

<!-- 配置商品被拦截器拦截 -->

<mvc:mapping path=*"/item/\*\*"* />

<!-- 配置具体的拦截器 -->

<bean class=*"com.igeek.ssm.interceptor.UserInterceptor"* />

</mvc:interceptor>

</mvc:interceptors>

# 关卡3

## 训练案例1

### 训练描述

参考前一天的案例实现如下功能

1. 实现商品维护人员多选，使用数组接受维护人id

2. 实现维护人批量提交，提交后台使用List接收人员数据

3. 改造搜索维护人功能，实现只能接受GET请求和POST请求

4. 修改维护人功能中，添加维护人照片上传功能

5. 在修改维护人成功后，跳转到维护人修改页面

6. 实现用户登录拦截，如果用户未登录，跳转到用户登录页

7. 实现权限拦截，账号和密码必须是admin才可以修改用户，其他用户只能查看用户列表