

淘淘商城

第九天

**讲师：传智.入云龙**

# 课程计划

* 商品详情页实现

1、商品查询服务事项

2、商品详情展示

3、添加缓存

# 实现商品详情页功能

## 功能分析

1、Taotao-portal接收页面请求，接收到商品id。

2、调用taotao-rest提供的商品详情的服务，把商品id作为参数传递给服务。接收到商品详细详细。

3、渲染结果，展示商品详细页面

4、为了提高响应速度，商品详情页的内容需要分步骤加载。

第一步：先展示商品基本信息，例如商品的名称、图片、价格。

第二步：展示商品的描述，此时商品页面已经展示完毕，需要ajax请求商品描述。展示的时机是页面加载完毕后一秒钟。

第三步：展示商品规格。当用户点击商品规格选项卡时展示。使用ajax请求商品规格。

5、在搜索页面点击商品的图片或者商品名称请求商品详情页面。

商品详情页请求的url：/item/{itemId}.html

商品描述请求的url：/item/desc/{itemId}.html

商品规格请求的url：/item/param/{itemId}.html

## 处理流程



## Taotao-rest服务

需要实现三个服务：

1. 根据商品id查询商品详细表。
2. 根据商品id查询商品描述表
3. 根据商品id查询商品规格参数表。

### Mapper

因为都是单表查询所以使用逆向工程生成的mapper文件即可。

tb\_item

tb\_item\_desc

tb\_item\_param\_item

### Service

|  |
| --- |
| @Service  **public** **class** ItemServiceImpl **implements** ItemService {  @Autowired  **private** TbItemMapper itemMapper;  @Autowired  **private** TbItemDescMapper itemDescMapper;  @Autowired  **private** TbItemParamItemMapper itemParamItemMapper;  /\*\*  \* 根据id取商品信息  \* <p>Title: getItemById</p>  \* <p>Description: </p>  \* **@param** id  \* **@return**  \* **@throws** Exception  \* **@see** com.taotao.rest.service.ItemService#getItemById(java.lang.Long)  \*/  @Override  **public** TbItem getItemById(Long id) **throws** Exception {  TbItem tbItem = itemMapper.selectByPrimaryKey(id);  **return** tbItem;  }  /\*\*  \* 根据id取商品描述  \* <p>Title: getItemDescById</p>  \* <p>Description: </p>  \* **@param** id  \* **@return**  \* **@throws** Exception  \* **@see** com.taotao.rest.service.ItemService#getItemDescById(java.lang.Long)  \*/  @Override  **public** TbItemDesc getItemDescById(Long id) **throws** Exception {  TbItemDesc itemDesc = itemDescMapper.selectByPrimaryKey(id);    **return** itemDesc;  }  /\*\*  \* 根据商品id取规格参数  \* <p>Title: getItemParamById</p>  \* <p>Description: </p>  \* **@param** id  \* **@return**  \* **@throws** Exception  \* **@see** com.taotao.rest.service.ItemService#getItemParamById(java.lang.Long)  \*/  @Override  **public** TbItemParamItem getItemParamById(Long id) **throws** Exception {  TbItemParamItemExample example = **new** TbItemParamItemExample();  Criteria criteria = example.createCriteria();  criteria.andItemIdEqualTo(id);  List<TbItemParamItem> list = itemParamItemMapper.selectByExampleWithBLOBs(example);  TbItemParamItem itemParamItem = **null**;  **if** (**null** !=**null** && !list.isEmpty()) {  itemParamItem = list.get(0);  }    **return** itemParamItem;    }  } |

### Controller

|  |
| --- |
| @Controller  @RequestMapping("/items")  **public** **class** ItemController {  @Autowired  **private** ItemService itemService;    @RequestMapping("/item/{id}")  @ResponseBody  **public** TaotaoResult getItemById(@PathVariable Long id) {  //有效性验证  **if** (id == **null**) {  **return** TaotaoResult.*build*(400, "参数中必须包含id");  }  TbItem tbItem = **null**;  //根据id查询商品信息  **try** {  tbItem = itemService.getItemById(id);  } **catch** (Exception e) {  e.printStackTrace();  //发生异常时返回异常信息  **return** TaotaoResult.*build*(500, ExceptionUtil.*getStackTrace*(e));  }  **return** TaotaoResult.*ok*(tbItem);  }    @RequestMapping("/itemdesc/{id}")  @ResponseBody  **public** TaotaoResult getItemDescById(@PathVariable Long id) {  //有效性验证  **if** (id == **null**) {  **return** TaotaoResult.*build*(400, "参数中必须包含id");  }  TbItemDesc tbItemDesc = **null**;  //根据id查询商品明细信息  **try** {  tbItemDesc = itemService.getItemDescById(id);  } **catch** (Exception e) {  e.printStackTrace();  //发生异常时返回异常信息  **return** TaotaoResult.*build*(500, ExceptionUtil.*getStackTrace*(e));  }  **return** TaotaoResult.*ok*(tbItemDesc);  }      @RequestMapping("/itemparam/{id}")  @ResponseBody  **public** TaotaoResult getItemParamById(@PathVariable Long id) {  //有效性验证  **if** (id == **null**) {  **return** TaotaoResult.*build*(400, "参数中必须包含id");  }  TbItemParamItem tbItemParamItem = **null**;  //根据id查询商品规格参数信息  **try** {  tbItemParamItem = itemService.getItemParamById(id);  } **catch** (Exception e) {  e.printStackTrace();  //发生异常时返回异常信息  **return** TaotaoResult.*build*(500, ExceptionUtil.*getStackTrace*(e));  }  **return** TaotaoResult.*ok*(tbItemParamItem);  }  } |

## Portal商品详情页实现

### 商品POJO

由于页面展示商品信息时，需要展示图片列表。多张图片在数据库中存储的格式是存储在同一个字段中使用逗号分隔的。所以商品展示时需要在pojo中做处理。故此在portal中自定义一个商品的pojo类。

|  |
| --- |
| **public** **class** Item {  **private** Long id;  **private** String title;  **private** String sellPoint;  **private** Long price;  **private** Integer num;  **private** String barcode;  **private** String image;  **private** Long cid;  **private** Byte status;  **private** Date created;  **private** Date updated;  **//省略get、set方法。。。。。。**  //添加此方法拆分图片列表  **public** String[] getImages() {  **if** (image != **null** && image != "") {  String[] strings = image.split(",");  **return** strings;  }  **return** **null**;  }  } |

### Service

|  |
| --- |
| @Service  **public** **class** ItemServiceImpl **implements** ItemService {  @Value("${REST\_BASE\_URL}")  **private** String REST\_BASE\_URL;  @Value("${ITEMS\_ITEM\_URL}")  **private** String ITEMS\_ITEM\_URL;  @Value("${ITEMS\_ITEMDESC\_URL}")  **private** String ITEMS\_ITEMDESC\_URL;  @Value("${ITEMS\_ITEMPARAM\_URL}")  **private** String ITEMS\_ITEMPARAM\_URL;  @Override  **public** Item getItemById(Long id) **throws** Exception {  // 查询商品信息  String result = HttpClientUtil.*doGet*(REST\_BASE\_URL + ITEMS\_ITEM\_URL + id);  // 转换成java对象  TaotaoResult taotaoResult = TaotaoResult.*formatToPojo*(result, Item.**class**);  Item item = **null**;  **if** (taotaoResult.getStatus() == 200) {  item = (Item) taotaoResult.getData();  }  **return** item;  }  @Override  **public** TbItemDesc geTbItemDescById(Long id) **throws** Exception {  // 查询商品信息  String result = HttpClientUtil.*doGet*(REST\_BASE\_URL + ITEMS\_ITEMDESC\_URL + id);  // 转换成java对象  TaotaoResult taotaoResult = TaotaoResult.*formatToPojo*(result, TbItemDesc.**class**);  TbItemDesc itemDesc = **null**;  **if** (taotaoResult.getStatus() == 200) {  itemDesc = (TbItemDesc) taotaoResult.getData();  }  **return** itemDesc;  }  @Override  **public** String geTbItemParamItemById(Long id) **throws** Exception {  // 查询商品信息  String result = HttpClientUtil.*doGet*(REST\_BASE\_URL + ITEMS\_ITEMPARAM\_URL + id);  // 转换成java对象  TaotaoResult taotaoResult = TaotaoResult.*formatToPojo*(result, TbItemParamItem.**class**);  String resultHtml = "";  **if** (taotaoResult.getStatus() == 200) {  **try** {  TbItemParamItem itemParamItem = (TbItemParamItem) taotaoResult.getData();  //取规格参数信息  String paramData = itemParamItem.getParamData();  //把规格参数转换成java对象  List<Map> paramList = JsonUtils.*jsonToList*(paramData, Map.**class**);  //拼装html  resultHtml ="<table cellpadding=\"0\" cellspacing=\"1\" width=\"100%\" border=\"0\" class=\"Ptable\">\n" +  " <tbody>\n";  **for** (Map map : paramList) {  resultHtml +=  " <tr>\n" +  " <th class=\"tdTitle\" colspan=\"2\">"+map.get("group")+"</th>\n" +  " </tr>\n";  List<Map> params = (List<Map>) map.get("params");  **for** (Map map2 : params) {    resultHtml +=  " <tr>\n" +  " <td class=\"tdTitle\">"+ map2.get("k")+"</td>\n" +  " <td>"+map2.get("v")+"</td>\n" +  " </tr>\n" ;  }    }  resultHtml += " </tbody>\n" +  "</table>";  } **catch** (Exception e){  //如果转换发送异常，忽略。返回一个空字符串。  e.printStackTrace();  }  }  **return** resultHtml;  }  } |

### Controller

|  |
| --- |
| @Controller  **public** **class** ItemController {  @Autowired  **private** ItemService itemService;    @RequestMapping("/item/{id}")  **public** String showItem(@PathVariable Long id, Model model) **throws** Exception {  //取商品信息  Item item = itemService.getItemById(id);  //把结果传递给页面  model.addAttribute("item", item);  //返回逻辑视图  **return** "item";  }  @RequestMapping(value="/item/desc/{id}", produces=MediaType.***TEXT\_HTML\_VALUE***+";charset=utf-8")  @ResponseBody  **public** String showItemDesc(@PathVariable Long id) **throws** Exception {  //取商品描述  TbItemDesc itemDesc = itemService.geTbItemDescById(id);  //返回商品描述信息  **return** itemDesc.getItemDesc();  }    @RequestMapping(value="/item/param/{id}", produces=MediaType.***TEXT\_HTML\_VALUE***+";charset=utf-8")  @ResponseBody  **public** String showItemParam(@PathVariable Long id) **throws** Exception {  //取规格参数  String itemParamItem = itemService.geTbItemParamItemById(id);  //返回规格参数信息  **return** itemParamItem;  }  } |

### 效果：



## 添加缓存逻辑

|  |
| --- |
| @Override  **public** TbItem getItemById(Long id) **throws** Exception {  //缓存中命中  //在redis中无法对hash中的可以做expire。所以使用另外一种方法：key的命名方法为“主key：id”  String itemCache = jedisCluster.get(TB\_ITEM\_KEY + ":" + id);  **try** {  **if** (!StringUtils.*isBlank*(itemCache)) {  TbItem tbItem = JsonUtils.*jsonToPojo*(itemCache, TbItem.**class**);  **return** tbItem;  }  } **catch** (Exception e) {  e.printStackTrace();  }  //如果缓存中没有数据，查询数据库  TbItem tbItem = itemMapper.selectByPrimaryKey(id);  //把数据缓存起来  **try** {  jedisCluster.set(TB\_ITEM\_KEY + ":" + id, JsonUtils.*objectToJson*(tbItem));  //设置过期时间，有效期一天  jedisCluster.expire(TB\_ITEM\_KEY + ":" + id, 60\*60\*24);  } **catch** (Exception e) {  e.printStackTrace();  }  **return** tbItem;  } |

其他方法课后作业。

# 缓存同步

Redis是内存数据库也属于稀缺资源，所以不应该永久占用，所以要设置过期时间。当商品内容更新后，需要同步缓存。同步方法参见内容部分的缓存同步方法。