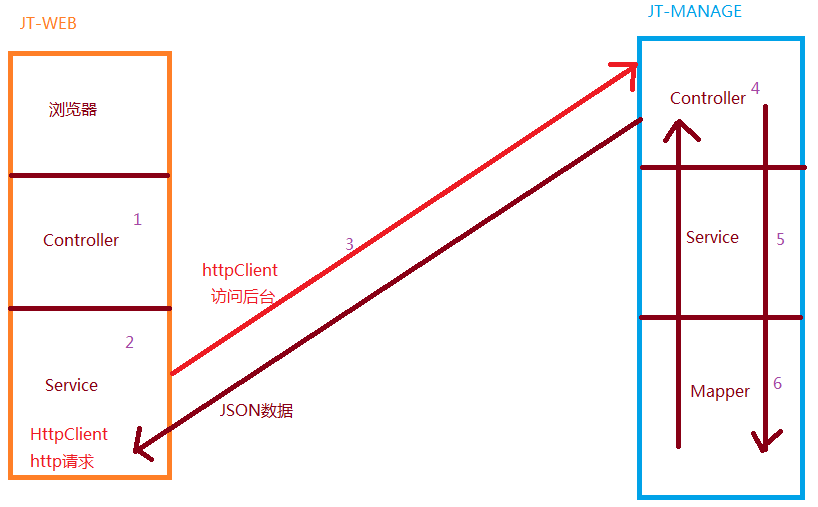
# 商品信息回显

## HttpClient调用流程

### 图解



## 编辑JT-manage获取商品数据

### 业务需求

由于jt-web需要进行数据展现.则需要返回商品信息以及商品描述信息.

规定url地址:

1. 查询商品信息:

<http://manage.jt.com/web/item/findItemById/123123>

1. 查询商品详情信息:

<http://manage.jt.com/web/item/findItemDescById/123123>

### 编辑后台Controller

@RestController

@RequestMapping("/web/item")

**public** **class** WebItemController {

//我是后台的Controller

@Autowired

**private** ItemService itemService;

//查询商品信息

@RequestMapping("/findItemById/{itemId}")

@Cache\_Find(keyType = KEY\_ENUM.***AUTO***)

**public** Item findItemById(@PathVariable Long itemId) {

**return** itemService.findItemById(itemId);

}

//查询商品详情信息

@RequestMapping("/findItemDescById/{itemId}")

@Cache\_Find(keyType = KEY\_ENUM.***AUTO***)

**public** ItemDesc findItemDescById(@PathVariable Long itemId) {

**return** itemService.findItemDescById(itemId);

}

}

### 编辑后台Service

@Override

**public** ItemDesc findItemDescById(Long itemId) {

**return** itemDescMapper.selectById(itemId);

}

@Override

**public** Item findItemById(Long itemId) {

**return** itemMapper.selectById(itemId);

}

### 访问测试





### pojo对象数据封装问题

<form id=*"cartForm"* method=*"post"*>

<input class=*"text"* id=*"buy-num"* name=*"num"* value=*"1"* onkeyup="setAmount.modify('#buy-num');"/>

<input type=*"hidden"* class=*"text"* name=*"itemTitle"* value=*"*${item.title }*"*/>

<input type=*"hidden"* class=*"text"* name=*"itemImage"* value=*"***${item.images[0]}***"*/>

<input type=*"hidden"* class=*"text"* name=*"itemPrice"* value=*"*${item.price}*"*/>

</form>

说明:页面在获取数据时调用**${item.images[0]}**通过pojo数据的get方法获取数据.但是item对象中没有images属性.所以利用如下配置补全代码

@TableName("tb\_item")

@Data

@Accessors(chain = **true**)

//如果get/set方法不全,添加该注解忽略转化.

**@JsonIgnoreProperties(ignoreUnknown = true)**

**public** **class** Item **extends** BasePojo{

@TableId(type = IdType.***AUTO***)

**private** Long id; //定义主键

**private** String title; //标题

**private** String sellPoint; //卖点信息

**private** Long price; //商品价格 1.dubbo有精度问题 0.88888..+0.1111112=0.9999.... //2.速度 int > long > dubbo 将商品扩大100倍保存

**private** Integer num; //商品数量

**private** String barcode; //二维码

**private** String image; //商品品图片信息 1.jpg,2.jpg,3.jpg

**private** Long cid; //商品分类信息

**private** Integer status; //商品状态 1正常，2下架，3删除'

//el表达式获取数据时调用对象的get方法...

**public** String[] getImages() {

**return** image.split(",");

}

}

### 前台页面分析



### 编辑前台Controller

@Controller

@RequestMapping("/items")

**public** **class** ItemController {

@Autowired

**private** ItemService itemService;

/\*\*

\* 编辑jt-web前台服务器.

\* 从后台manage中获取数据,之后页面展现.

\*/

@RequestMapping("/{itemId}")

**public** String findItemById(@PathVariable Long itemId,Model model) {

Item item = itemService.findItemById(itemId);

//将item对象保存到request域中,方便el表达式获取数据

model.addAttribute("item", item);

**return** "item"; //跳转的页面!!!!

}

}

### 编辑Service

@Service

**public** **class** ItemServiceImpl **implements** ItemService {

@Autowired

**private** HttpClientService httpClient;

@Override

**public** Item findItemById(Long itemId) {

String url = "http://manage.jt.com/web/item/findItemById/"+itemId;

String itemJSON = httpClient.doGet(url);

**return** ObjectMapperUtil.*toObject*(itemJSON,Item.**class**);

}

}

### 页面展现



### 关于JSON转化问题

需求:页面中必须获取一个images[0]数组!!!

数据准备:item对象中必须满足要求

1. images属性
2. **getImages();**

问题:

json转化时必然报错.json转化对象时没有setImages方法,所以必须报错.

**解决问题:**

@JsonIgnoreProperties(ignoreUnknown = **true**)

java.lang.RuntimeException: com.fasterxml.jackson.databind.exc.UnrecognizedPropertyException: **Unrecognized field "images"** (class com.jt.pojo.Item), not marked as ignorable (11 known properties: "barcode", "status", "num", "created", "price", "updated", "sellPoint", "title", "id", "cid", "image"])

at [Source: (String)"{"created":"2019-07-08T06:03:54.000+0000","updated":"2019-07-08T06:03:54.000+0000","id":1474391984,"title":"京东超市 汽车大冒险闯关轨道桌面益智儿童益智玩具亲自互动创意停车场3-6岁火车套装 闯关大冒险【3只汽车+2只火车】","sellPoint":"闯关大冒险【3只汽车+2只火车】限时秒杀价149元，7月8号14点开抢，八大关卡玩个够","price":14900,"num":100,"barcode":"123123","image":"http://image.jt.com/2019/07/08/f8032e9832db4d1d884911195be91fc9.jpg,http://image.jt.com/2019/07/08/3c141a97f1f84c1e8ec0a77090c88800.png,http://image.jt.com/2019/07/08/ed53a5d393594dc5b873c5221acb4697.jpg","cid":497,"status"[truncated 225 chars]; line: 1, column: 515] (through reference chain: com.jt.pojo.Item["images"])

## 实现商品详情查询

### 页面分析

需要展现商品详情数据.利用el表达式从域中取值.

<div class="detail-content">

${itemDesc.itemDesc }

</div>

### 编辑前台Controller

/\*\*

\* 编辑jt-web前台服务器.

\* 从后台manage中获取数据,之后页面展现.

\*/

@RequestMapping("/{itemId}")

**public** String findItemById(@PathVariable Long itemId,Model model) {

Item item = itemService.findItemById(itemId);

ItemDesc itemDesc = itemService.findItemDescById(itemId);

//将item对象保存到request域中,方便el表达式获取数据

model.addAttribute("item", item);

model.addAttribute("itemDesc", itemDesc);

**return** "item"; //跳转的页面!!!!

}

### 编辑Service

@Override

**public** ItemDesc findItemDescById(Long itemId) {

String url = "http://manage.jt.com/web/item/findItemDescById/"+itemId;

String itemDescJSON = httpClient.doGet(url);

**return** ObjectMapperUtil.*toObject*(itemDescJSON, ItemDesc.**class**);

}

# 用户单点登录业务

## 页面通用跳转

### 业务需求

1. url: <http://www.jt.com/user/login.html>
2. url: <http://www.jt.com/user/register.html>

说明:根据用户请求.在jt-web中实现跳转页面.

### 编辑Controller

@Controller

@RequestMapping("/user")

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

//通用页面跳转

@RequestMapping("/{moduleName}")

**public** String moduleName(@PathVariable String moduleName) {

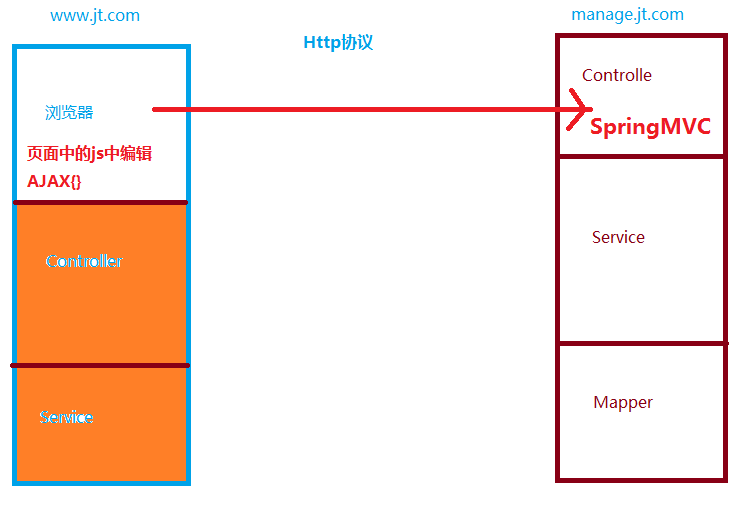
**return** moduleName;

}

}

## 跨域的实现

### 代码优化策略



说明:由于采用httpClient调用方式,程序代码的层级较多.编码的效率低.

代码能否优化.能否利用浏览器解析js发起ajax请求实现远程数据的访问???

### 远程跨域访问测试

#### 商品后台测试

通过<http://manage.jt.com/test.html>完全获取正确的数据

<script type="text/javascript" src="http://manage.jt.com/js/jquery-easyui-1.4.1/jquery.min.js"></script>

<script type="text/javascript">

$(**function**(){

$.get("http://manage.jt.com/test.json",**function**(data){

alert(data.id);

alert(data.name);

})

})

</script>

#### 在jt-web中测试远程调用

<http://www.jt.com/test.html>,发起请求时发现调用不能正常进行.

<script type="text/javascript" src="http://manage.jt.com/js/jquery-easyui-1.4.1/jquery.min.js"></script>

<script type="text/javascript">

/\*$(){}结构必然是jQuery函数类库导入后才能正确执行\*/

$(**function**(){

alert("我执行了AJAX");

//利用jQuery发起AJAX请求

$.get("http://manage.jt.com/test.json",**function**(data){

alert(data.id);

alert(data.name);

})

})

</script>

### 关于浏览器同源策略问题

同源策略:

浏览器在解析js时发起ajax中有要求.

如果 **协议://域名:端口** 都相同则满足同源策略的要求.浏览器可以实现正常通信.

如果上述的三项有一项不满足,则**违反了同源策略**.浏览器可以正确的发起请求.也可以正确的获取返回值结果.但是浏览器**不予解析返回值**!!!

总结:浏览器出于安全性考虑.必须满足同源策略.

### 同源策略案例



例子:

manage.jt.com/test.json

localhost:8091/test2.json

问题1:域名不同

2:端口不同 80 8091

### 跨域实现原理

需求:业务中经常会遇到A服务器访问B服务器的数据.但是由于同源策略的约束导致请求不能正确的运行. 如何实现跨域????

解决思路:

1. 利用javaScript中的**src开放原则** 实现跨域!!!!
2. 自定义回调函数
3. 将返回值进行特殊的格式封装 callback(返回值数据)

<!DOCTYPE html>

<html>

<head>

<meta charset=*"UTF-8"*>

<title>测试JSON跨域问题</title>

<script type=*"text/javascript"*>

/\*JS是解释执行的语言 \*/

/\*定义回调函数 \*/

**function** hello(data){

alert(data.id);

alert(data.name);

}

</script>

<!--该json一直保存到浏览器中等待调用,但是没有函数名称无法调用 -->

<script type=*"text/javascript"* src=*"http://manage.jt.com/test.json"*></script>

<script type=*"text/javascript"* src=*"http://manage.jt.com/js/jquery-easyui-1.4.1/jquery.min.js"*></script>

</head>

<body>

<h1>JS跨域问题</h1>

</script>

</body>

</html>

返回值数据封装:

hello({"id":*"1"*,"name":*"tom"*})

### JSONP介绍

JSONP(JSON with Padding)是[JSON](https://baike.baidu.com/item/JSON)的一种“使用模式”，**可用于解决主流浏览器的跨域数据访问的问题**。由于同源策略，一般来说位于 server1.example.com 的网页无法与不是 server1.example.com的服务器沟通，而 HTML 的<script> 元素是一个例外。利用 <script> 元素的这个**开放策略**，网页可以得到从其他来源动态产生的 JSON 资料，而这种使用模式就是所谓的 JSONP。用 JSONP 抓到的资料并不是 JSON，而是任意的JavaScript，用 JavaScript 直译器执行而不是用 JSON 解析器解析。

## jQuery实现JSONP的调用

### 编辑jQueryAjax

<script type="text/javascript">

$(**function**(){

alert("测试访问开始!!!!!")

$.ajax({

url:"http://manage.jt.com/web/testJSONP",

type:"get", //jsonp只能支持get请求

dataType:"jsonp", //dataType表示返回值类型

//jsonp: "callback", //指定参数名称

//jsonpCallback: "hello", //指定回调函数名称

success:**function** (data){ //data经过jQuery封装返回就是json串

alert(data.id);

alert(data.name);

//转化为字符串使用

//var obj = eval("("+data+")");

//alert(obj.name);

}

});

})

</script>

### 编辑后台Controller

@RestController //要求返回json数据

**public** **class** JSONPController {

/\*

\* jsonp返回值结果,必须经过特殊格式封装.

\* 调用者::回调函数获取

\* 数据返回:封装数据 callback(JSON串)

\* http://manage.jt.com/web/testJSONP

\* ?callback=jQuery111108050409315062621\_1563263312902&\_=1563263312903

\*/

@RequestMapping("/web/testJSONP")

**public** String testJSONP(String callback) {

ItemCat itemCat = **new** ItemCat();

itemCat.setId(1000L).setName("jsonp测试!!!");

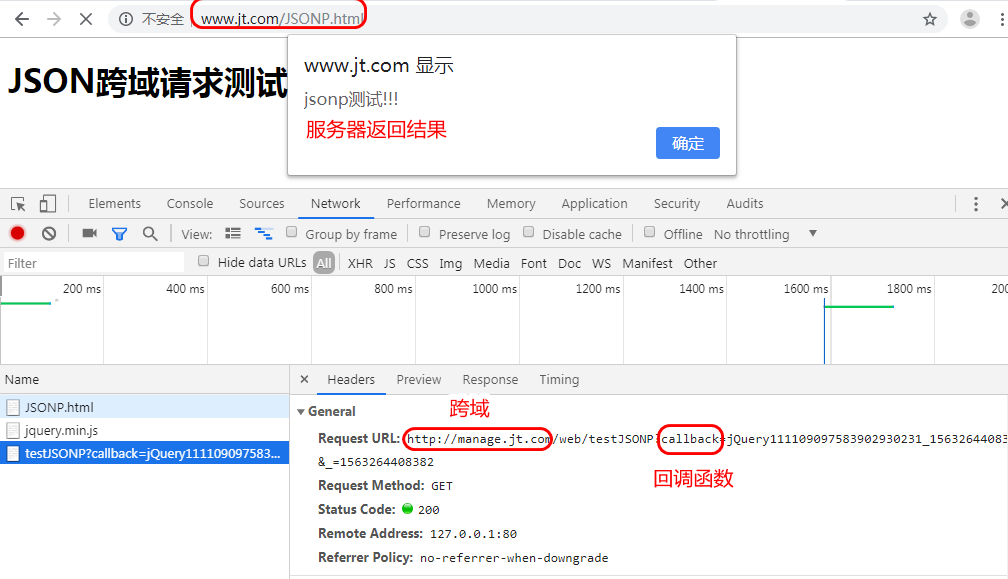
String json = ObjectMapperUtil.*toJSON*(itemCat);

**return** callback+"("+json+")";

}

}

### 跨域测试



### JSONP高级API

/\*\*

\* 参数说明:

\* function:代表回调函数名称

\* value: 代表服务器返回的数据

\*

\* **@param** callback

\* **@return**

\*/

@RequestMapping("/web/testJSONP")

**public** JSONPObject testJSONP(String callback) {

ItemCat itemCat = **new** ItemCat();

itemCat.setId(2000L).setName("JSONP简化写法!!!");

JSONPObject object = **new** JSONPObject(callback, itemCat);

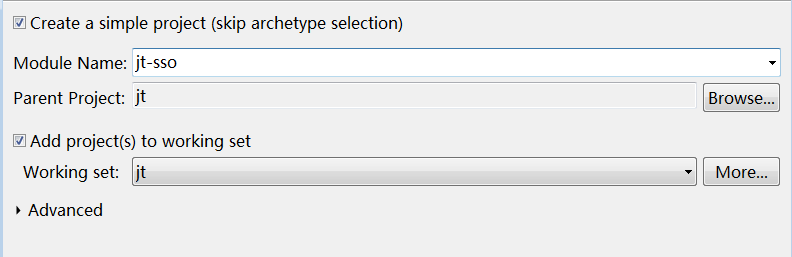
**return** object;

}

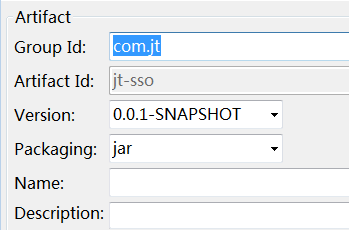
## 单点登录系统搭建

### 创建项目

1.定义名称



2.选择jar包



### 添加继承和依赖

<parent>

<groupId>com.jt</groupId>

<artifactId>jt</artifactId>

<version>0.0.1-SNAPSHOT</version>

</parent>

<artifactId>jt-sso</artifactId>

<dependencies>

<dependency>

<groupId>com.jt</groupId>

<artifactId>jt-common</artifactId>

<version>0.0.1-SNAPSHOT</version>

</dependency>

</dependencies>

<build>

<plugins>

<!--跳过测试类打包 -->

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-surefire-plugin</artifactId>

<configuration>

<skip>true</skip>

</configuration>

</plugin>

<!--表示maven打包的插件 -->

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

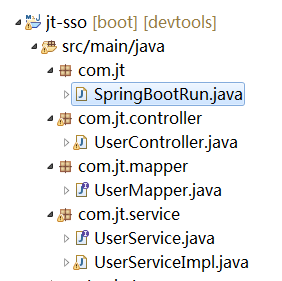
</plugins>

</build>

### 导入src文件

将课前资料中的jt-sso的src文件导入.

D:\1-JT\每日课前资料\day10-day11-京淘前台搭建-httpclient实现数据获取-jsonp实现跨域\2-跨域实现\1-jt-sso项目搭建\jt-sso



修改数据如下:

1. 修改redis.properties
2. 删除spring配置文件
3. 编辑POJO编辑Mapper接口

### 编辑POJO对象

@Data

@Accessors(chain = **true**)

@TableName("tb\_user")

**public** **class** User **extends** BasePojo{

@TableId(type = IdType.***AUTO***)

**private** Long id;

**private** String username;

**private** String password;

**private** String phone;

**private** String email;

}

### 编辑Mapper接口

**public** **interface** UserMapper **extends** BaseMapper<User>{

}

### 作业:实现用户数据校验

|  |  |
| --- | --- |
| 请求方法 | GET |
| URL | http://sso.jt.com/user/check/{param}/{type} |
| 参数 | 格式如：chenchen/1  其中chenchen是校验的数据  Type为类型，可选参数1 username、2 phone、3 email |
| 示例 | http://sso.jt.com/user/check/chenchen/1 |
| 返回值 | {  status: 200 //200 成功，201 没有查到  msg: “OK” //返回信息消息  data: false //返回数据true用户已存在，false用户不存在，可以  } |
|  |  |