# 京淘单点登录

## 用户信息回显

### 页面分析

1.页面url



2.页面js

$.ajax({

url : "http://sso.jt.com/user/query/" + \_ticket,

dataType : "jsonp",

type : "GET",

success : **function**(data){

**if**(data.status == 200){

**var** \_data = JSON.parse(data.data);

**var** html =\_data.username+"，欢迎来到京淘！<a href=\"http://www.jt.com/user/logout.html\" class=\"link-logout\">[退出]</a>";

$("#loginbar").html(html);

}

}

});

### 编辑Controller

//通过跨域,实现用户信息的获取

@RequestMapping("/query/{token}")

**public** JSONPObject findUserByToken(String callback,

@PathVariable String token) {

String userJSON = jedisCluster.get(token);

SysResult sysResult = SysResult.*oK*(userJSON);

**return** **new** JSONPObject(callback, sysResult);

}

## 用户退出

### 编辑Controller

//实现用户退出操作 1.删除cookie 2.删除redis

@RequestMapping("/logout")

**public** String logout(HttpServletRequest request,HttpServletResponse response) {

Cookie[] cookies = request.getCookies();

String token = **null**;

**for** (Cookie cookie : cookies) {

**if**("JT\_TICKET".equals(cookie.getName())){

token = cookie.getValue();

**break**;

}

}

//删除redis

jedisCluster.del(token);

//删除cookie

Cookie cookie = **new** Cookie("JT\_TICKET","");

cookie.setMaxAge(0);

cookie.setPath("/");

response.addCookie(cookie);

//执行完成后,将页面跳转到系统首页

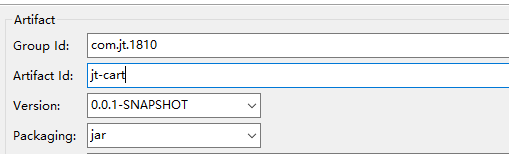
**return** "redirect:/index.html";

}

# SpringBoot整合购物车

## 创建jt-cart项目

### 新建项目



### 编辑POM.xml文件

<parent>

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

<artifactId>spring-boot-starter-parent</artifactId>

<version>2.1.3.RELEASE</version>

<relativePath /> <!-- lookup parent from repository -->

</parent>

<properties>

<java.version>1.8</java.version>

</properties>

<dependencies>

<!--springBoot动态的引入springMVC全部的配置 -->

<dependency>

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

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<!--引入测试类 -->

<dependency>

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

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

<!--添加属性注入依赖 -->

<dependency>

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

<artifactId>spring-boot-configuration-processor</artifactId>

<optional>true</optional>

</dependency>

<!--支持热部署 -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>springloaded</artifactId>

<version>1.2.8.RELEASE</version>

</dependency>

<dependency>

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

<artifactId>spring-boot-devtools</artifactId>

</dependency>

<!--引入插件lombok 自动的set/get/构造方法插件 -->

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

</dependency>

<!--引入数据库驱动 -->

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<scope>runtime</scope>

</dependency>

<!--引入druid数据源 -->

<dependency>

<groupId>com.alibaba</groupId>

<artifactId>druid</artifactId>

<version>1.1.12</version>

</dependency>

<!--spring整合mybatis-plus -->

<dependency>

<groupId>com.baomidou</groupId>

<artifactId>mybatis-plus-boot-starter</artifactId>

<version>3.0.6</version>

</dependency>

<!--spring整合redis -->

<dependency>

<groupId>redis.clients</groupId>

<artifactId>jedis</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.data</groupId>

<artifactId>spring-data-redis</artifactId>

</dependency>

</dependencies>

<build>

<plugins>

<!--引入springBoot项目启动项 内置tomcat -->

<plugin>

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

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

</plugin>

</plugins>

</build>

### 编辑POJO对象

@Data

@Accessors(chain=**true**)

@TableName("tb\_cart")

**public** **class** Cart **extends** BasePojo{

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

**private** Long id;

**private** Long userId;

**private** Long itemId;

**private** String itemTitle;

**private** String itemImage;

**private** Long itemPrice;

**private** Integer num;

}

### 编辑Mapper接口/映射文件

**public** **interface** CartMapper **extends** BaseMapper<Cart>{

}

编辑映射文件

<?xml version=*"1.0"* encoding=*"UTF-8"* ?>

<!DOCTYPE mapper

PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"

"http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace=*"com.jt.cart.mapper.CartMapper"*>

</mapper>

### 编辑yml文件

server:

port: 8094

servlet:

context-path: /

spring:

datasource:

#引入druid数据源

type: com.alibaba.druid.pool.DruidDataSource

driver-class-name: com.mysql.jdbc.Driver

url: jdbc:mysql://127.0.0.1:3306/jtdb?serverTimezone=GMT%2B8&useUnicode=true&characterEncoding=utf8&autoReconnect=true&allowMultiQueries=true

username: root

password: root

#mybatis-plush配置

mybatis-plus:

type-aliases-package: **com.jt.cart.pojo**

mapper-locations: classpath:/mybatis/mappers/\*.xml

configuration:

map-underscore-to-camel-case: **true** #自动开启驼峰规则映射

### 配置nginx

#编辑购物车服务器

server {

listen 80;

server\_name cart.jt.com;

location / {

#反向代理到url

proxy\_pass http://localhost:8094;

}

}

#编辑购物车服务器

server {

listen 80;

server\_name order.jt.com;

location / {

#反向代理到url

proxy\_pass http://localhost:8095;

}

}

## 购物车列表展现

### 页面分析



### 编辑工具类

/\*\*

\* 该工具类实现objecjtMapper的对象的转化,

\* 优化了try-cache方法

\* **@author** LYJ

\*

\*/

**public** **class** MapperUtil {

//将对象转化为json写法

**public** **static** String toJSON(Object object) {

ObjectMapper mapper = **new** ObjectMapper();

String json = **null**;

**try** {

json = mapper.writeValueAsString(object);

} **catch** (JsonProcessingException e) {

e.printStackTrace();

**throw** **new** RuntimeException();

}

**return** json;

}

//将json数据转化为对象

**public** **static** <T> T toObject(String json,Class<?> targetClass){

ObjectMapper mapper = **new** ObjectMapper();

T object = **null**;

**try** {

object = (T) mapper.readValue(json, targetClass);

} **catch** (Exception e) {

e.printStackTrace();

**throw** **new** RuntimeException();

}

**return** object;

}

}

### 编辑前台Controller

@Controller

@RequestMapping("/cart")

**public** **class** CartController {

@Autowired

**private** CartService cartService;

//实现购物车列表展现

@RequestMapping("/show")

**public** String findCartList(Model model) {

Long userId = 7L; //暂时写死 7L

List<Cart> cartList =

cartService.findCartListByUserId(userId);

model.addAttribute("cartList", cartList);

**return** "cart";

}

}

### 编辑前台Service

//操作jt-web

@Override

**public** List<Cart> findCartListByUserId(Long userId) {

String url = "http://cart.jt.com/cart/query/"+userId;

String result = httpClient.doGet(url);

SysResult sysResult =

MapperUtil.*toObject*(result,SysResult.**class**);

List<Cart> cartList =

(List<Cart>) sysResult.getData();

**return** cartList;

}

### 编辑后台Controller

@RestController

@RequestMapping("/cart")

**public** **class** CartController {

@Autowired

**private** CartService cartService;

//实现购物车列表查询

@RequestMapping("/query/{userId}")

**public** SysResult findCartListByUserId(@PathVariable Long userId) {

List<Cart> cartList =

cartService.findCartListByUserId(userId);

**return** SysResult.*oK*(cartList);

}

}

### 编辑后台Service

@Service

**public** **class** CartServiceImpl **implements** CartService {

@Autowired

**private** CartMapper cartMapper;

@Override

**public** List<Cart> findCartListByUserId(Long userId) {

QueryWrapper<Cart> queryWrapper =

**new** QueryWrapper<Cart>();

queryWrapper.eq("user\_id", userId);

**return** cartMapper.selectList(queryWrapper);

}

}

### 页面效果



## 购物车数量的修改

### 页面分析

1.页面url地址



2.页面JS

$(".increment").click(**function**(){//＋

**var** \_thisInput = $(**this**).siblings("input");

\_thisInput.val(eval(\_thisInput.val()) + 1);

$.post("/service/cart/update/num/"+\_thisInput.attr("itemId")+"/"+\_thisInput.val(),**function**(data){

TTCart.refreshTotalPrice();

});

});

### 编辑前台Controller

//编辑前台Controller

@RequestMapping("/update/num/{itemId}/{num}")

@ResponseBody

**public** SysResult updateCartNum(

@PathVariable Long itemId,

@PathVariable Integer num) {

**try** {

Cart cart = **new** Cart();

cart.setUserId(7L);

cart.setItemId(itemId);

cart.setNum(num);

cartService.updateCart(cart);

**return** SysResult.*oK*();

} **catch** (Exception e) {

e.printStackTrace();

}

**return** SysResult.*build*(201,"用户修改数量失败");

}

### 编辑前台Service

//实现购物车数量的修改

@Override

**public** **void** updateCart(Cart cart) {

String url = "http://cart.jt.com/cart/update/num/"

+cart.getUserId()

+"/"+ cart.getItemId()

+"/"+ cart.getNum();

httpClient.doGet(url);

}

### 编辑后台Controller

//修改购物车数量

@RequestMapping("/update/num/{userId}/{itemId}/{num}")

**public** SysResult updateCartNum(Cart cart) {

**try** {

cartService.updateCartNum(cart);

**return** SysResult.*oK*();

} **catch** (Exception e) {

e.printStackTrace();

}

**return** SysResult.*build*(201,"购物车修改失败");

}

### 编辑后台Service

@Override

**public** **void** updateCartNum(Cart cart) {

Cart cartTemp = **new** Cart();

cartTemp.setNum(cart.getNum())

.setUpdated(**new** Date());

UpdateWrapper<Cart> updateWrapper = **new** UpdateWrapper<>();

updateWrapper.eq("user\_id", cart.getUserId())

.eq("item\_id", cart.getItemId());

cartMapper.update(cartTemp, updateWrapper);

}

### 模拟测试

http://cart.jt.com/cart/update/num/7/562379/100



### 406报错

核心:如果页面中要求返回html代码,而返回的数据是json时,则浏览器会报解析异常406

正确测试:

<http://www.jt.com/service/cart/update/num/562379/5>

返回结果是JSON串

错误测试:

<http://www.jt.com/cart/update/num/562379/96.html>

返回结果是JSON串

因为以.html结尾.但是返回的数据是JSON.浏览器解析异常.

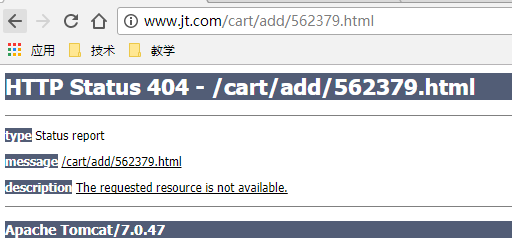
总结:

1. 为了增强搜索引擎的友好性采用伪静态的技术.以.html跳转页面.
2. 如果请求时AJAX.则数据请求时则添加/service/xxxxxxx

## 购物车新增

### 页面分析

1. url分析



1. js请求

//利用post传值

**function** addCart(){

**var** url = "http://www.jt.com/cart/add/${item.id}.html";

document.forms[0].action = url; //js设置提交链接

document.forms[0].submit(); //js表单提交

}

### 编辑前台Controller

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

**public** String saveCart(Cart cart) {

Long userId = 7L;

cart.setUserId(userId);

cartService.saveCart(cart);

**return** "redirect:/cart/show.html";

}

### 编辑前台Service

@Override

**public** **void** saveCart(Cart cart) {

String url = "http://cart.jt.com/cart/save";

String cartJSON = MapperUtil.*toJSON*(cart);

Map<String,String> params =

**new** HashMap<String, String>();

params.put("cartJSON",cartJSON);

httpClient.doPost(url, params);

}

### 编辑后台Controller

//实现购物车新增操作

@RequestMapping("/save")

**public** SysResult saveCart(String cartJSON) {

**try** {

Cart cart =

MapperUtil.*toObject*(cartJSON, Cart.**class**);

cartService.saveCart(cart);

**return** SysResult.*oK*();

} **catch** (Exception e) {

e.printStackTrace();

}

**return** SysResult.*build*(201,"购物车新增失败");

}

### 编辑后台Service

/\*\*

\* 新增业务逻辑 item\_id,user\_id

\* 如果根据item\_id和user\_id查询时

\* 数据库中有记录:

\* 则只做数据的数量的更新操作.

\* 如果数据库中没有记录:

\* 则做数据的新增操作.

\*/

@Override

@Transactional

**public** **void** saveCart(Cart cart) {

QueryWrapper<Cart> queryWrapper =

**new** QueryWrapper<>();

queryWrapper.eq("item\_id",cart.getItemId())

.eq("user\_id",cart.getUserId());

Cart cartDB =

cartMapper.selectOne(queryWrapper);

**if**(cartDB == **null**) {

cart.setCreated(**new** Date());

cart.setUpdated(cart.getCreated());

cartMapper.insert(cart);

}**else** {

**int** num = cart.getNum() + cartDB.getNum();

//只根据主键更新数据,并且全部字段更新!

Cart cartTemp = **new** Cart();

cartTemp.setNum(num);

UpdateWrapper<Cart> updateWrapper

= **new** UpdateWrapper<>();

updateWrapper

.eq("item\_id",cart.getItemId())

.eq("user\_id",cart.getUserId());

cartMapper.update(cartTemp,updateWrapper);

}

}