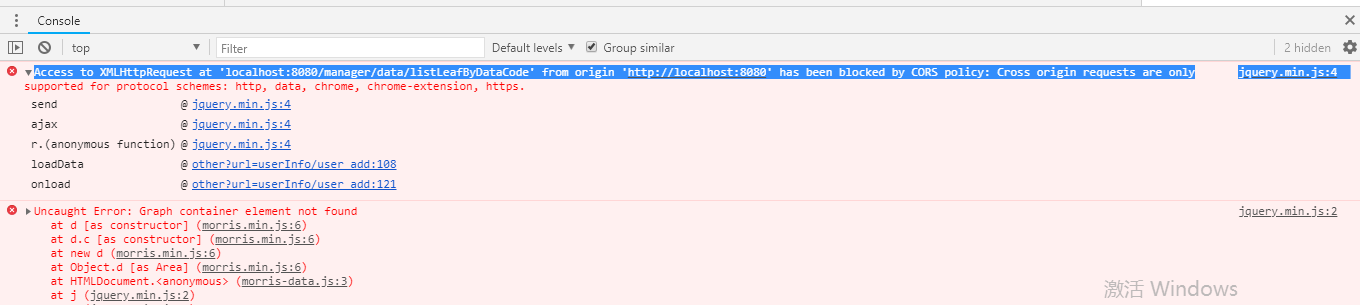
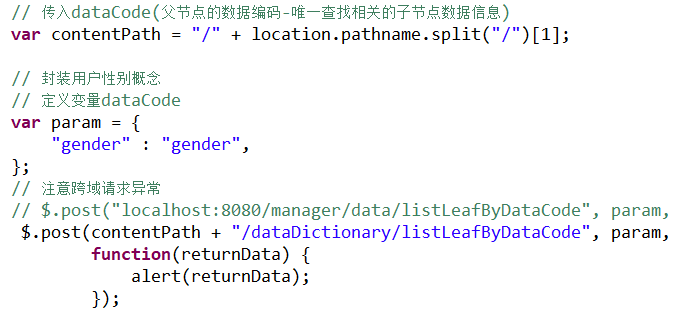
# 哒哒报表服务平台开发问题分析

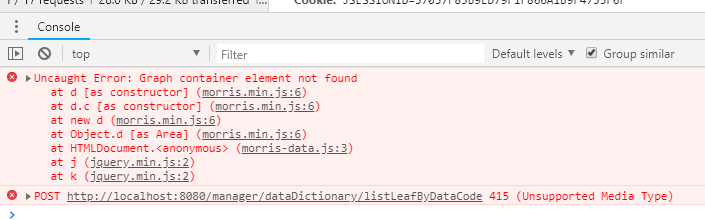
## 常见问题分析

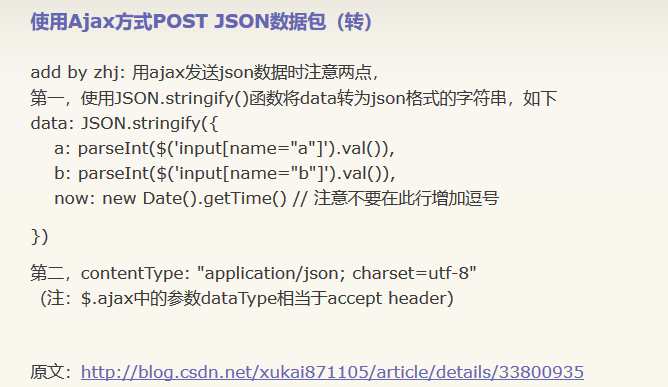
### 跨域请求失败



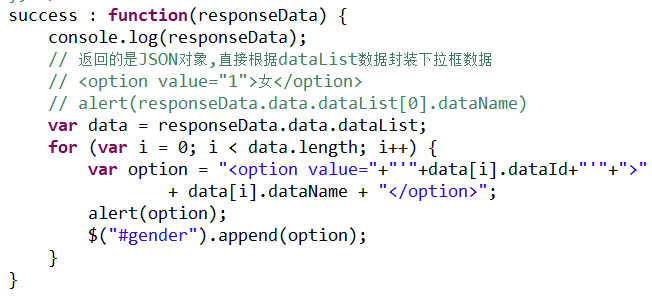


### 2.415错误





### 3.json下拉框数据封装



### 4.日期转化json异常

需要引入相关的jar并在实体类添加注解：



### 5.Bootstrap Validator前端数据验证

引入相关js、css文件

<!-- 新 Bootstrap 核心 CSS 文件 -->

<link href="http://cdn.static.runoob.com/libs/bootstrap/3.3.7/css/bootstrap.min.css" rel="stylesheet">

<link href="http://cdn.bootcss.com/bootstrap-validator/0.5.3/css/bootstrapValidator.min.css" rel="stylesheet" />

<link href="./index.css" rel="stylesheet">

<!-- jQuery文件。务必在bootstrap.min.js 之前引入 -->

<script src="http://cdn.static.runoob.com/libs/jquery/2.1.1/jquery.min.js"></script>

<!-- 最新的 Bootstrap 核心 JavaScript 文件 -->

<script src="http://cdn.static.runoob.com/libs/bootstrap/3.3.7/js/bootstrap.min.js"></script>

<script src="http://cdn.bootcss.com/bootstrap-validator/0.5.3/js/bootstrapValidator.min.js"></script>

在相关文件中验证前端数据

$(**function**() {

$("#submitForm").bootstrapValidator({

live : 'disabled',//验证时机，enabled是内容有变化就验证（默认），disabled和submitted是提交再验证

excluded : [ ':disabled', ':hidden', ':not(:visible)' ],//排除无需验证的控件，比如被禁用的或者被隐藏的

submitButtons : '#btn-test',//指定提交按钮，如果验证失败则变成disabled，但我没试成功，反而加了这句话非submit按钮也会提交到action指定页面

message : '通用的验证失败消息',//好像从来没出现过

feedbackIcons : {//根据验证结果显示的各种图标

valid : 'glyphicon glyphicon-ok',

invalid : 'glyphicon glyphicon-remove',

validating : 'glyphicon glyphicon-refresh'

},

fields : {

userName : {

validators : {

notEmpty : {//检测非空,radio也可用

message : '文本框必须输入'

},

stringLength : {//检测长度

min : 6,

max : 30,

message : '长度必须在6-30之间'

},

regexp : {//正则验证

regexp : /^[a-zA-Z0-9\_\.]+$/,

message : '所输入的字符不符要求'

},

remote : {//将内容发送至指定页面验证，返回验证结果，比如查询用户名是否存在

url : '指定页面',

message : 'The username is not available'

},

different : {//与指定文本框比较内容相同

field : '指定文本框name',

message : '不能与指定文本框内容相同'

},

emailAddress : {//验证email地址

message : '不是正确的email地址'

},

identical : {//与指定控件内容比较是否相同，比如两次密码不一致

field : 'confirmPassword',//指定控件name

message : '输入的内容不一致'

},

date : {//验证指定的日期格式

format : 'YYYY/MM/DD',

message : '日期格式不正确'

},

choice : {//check控件选择的数量

min : 2,

max : 4,

message : '必须选择2-4个选项'

}

}

}

}

});

// 如果是submit则直接验证，如果是button类型则调用验证

$("#submitButton").click(**function**() {//非submit按钮点击后进行验证，如果是submit则无需此句直接验证

$("#submitForm").bootstrapValidator('validate');//提交验证

**if** ($("#submitForm").data('bootstrapValidator').isValid()) {//获取验证结果，如果成功，执行下面代码

alert("yes");//验证成功后的操作，如ajax

}

});

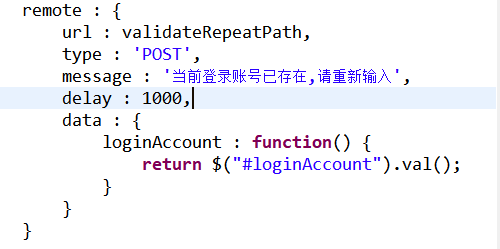
});

登录验证表达提交出错？

问题分析：在编辑信息时出现通过bootstrap validator校验Ajax请求提交两次问题

原因分析：可以看到remote那里有一个ajax验证重名（请求服务器后台数据进行验证），但实际处理效果是bootstrapValidator并没有等该ajax拿到返回值就直接获取一个默认值false从而导致该问题，可考虑在前端借助ajax回调验证时设置请求方式为同步（即async设置为非异步），等到remote验证完成获取到返回结果之后再进行下一步操作。或者是在交互上进行控制，不要在点击按钮时统一验证数据，而是通过当鼠标离开焦点的时候就触发验证，

或者设置延迟时间（无效）。



参考：datatables

<https://blog.csdn.net/wsjzzcbq/article/details/82348512>

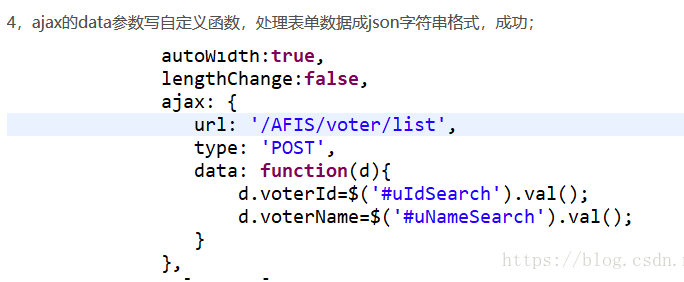
考虑多层级json数据处理

<https://blog.csdn.net/guoxinjie17/article/details/80342329>

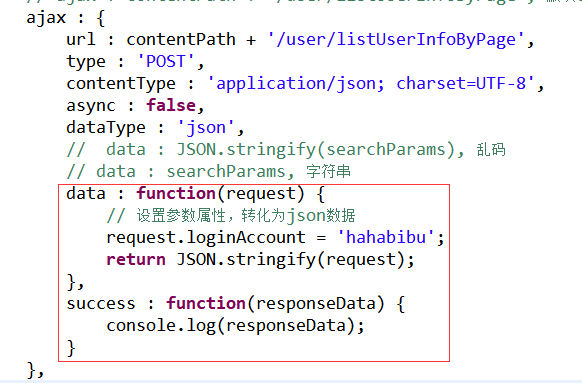
<https://www.cnblogs.com/zcy12590/p/7083451.html>

常见问题分析：

<https://stackoverflow.com/questions/25377637/datatables-cannot-read-property-mdata-of-undefined>



<https://blog.csdn.net/dongyuguoai/article/details/80785036>



<https://blog.csdn.net/xusen112/article/details/80910214>

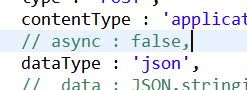
列与列数据要对应

Ajax请求成功后数据一直显示加载中：



<https://blog.csdn.net/qq_23315711/article/details/68921598>

同步警告：Synchronous XMLHttpRequest on the main thread is deprecated because of its d.....



存在问题:m目前数据如果为空无法封装到页面中？

时间戳转换

参考：<https://blog.csdn.net/java_mdzy/article/details/54970560>

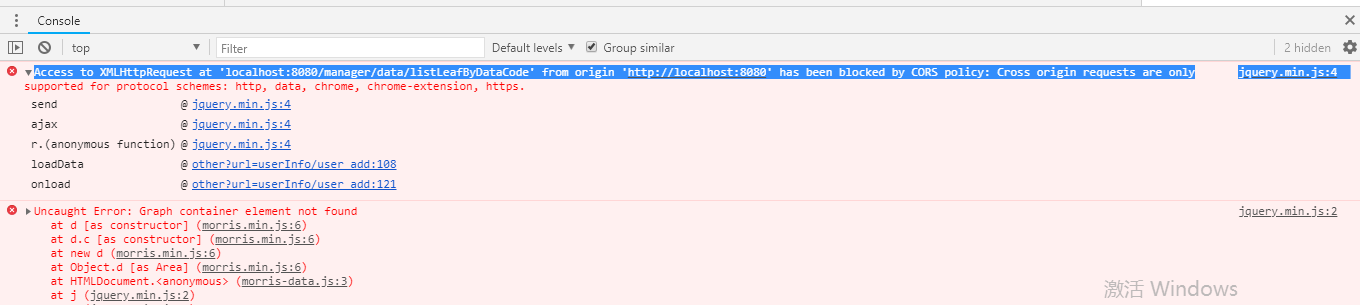
<https://www.cnblogs.com/SunAutumn/p/7800082.html>

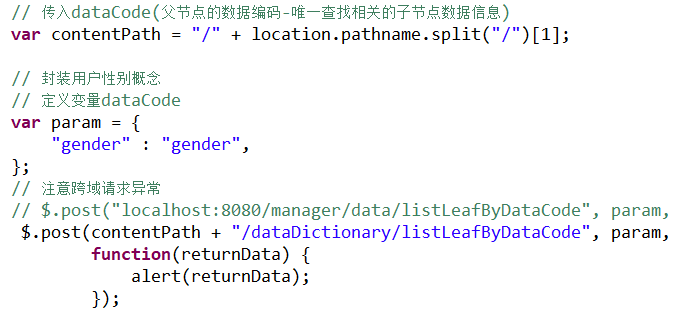
Ajax提交数据

async 设置为 false，则所有的请求均为同步请求，在没有返回值之前，同步请求将锁住浏览器，用户其它操作必须等待请求完成才可以执行。否则导致数据还没请求完成表完成数据提交和其他验证了

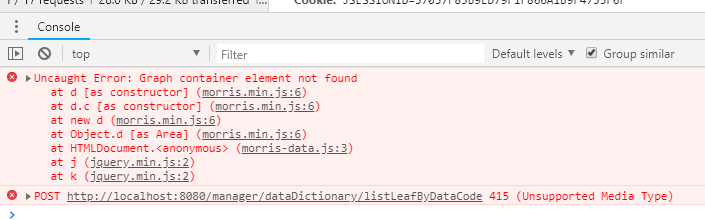
<https://blog.csdn.net/u013380777/article/details/52067483>

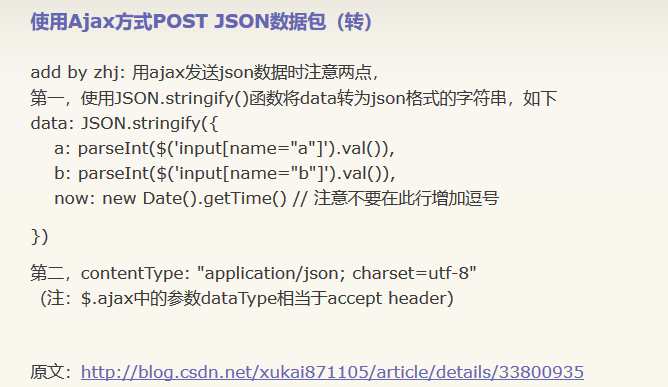
1. Column属性跨域请求失败

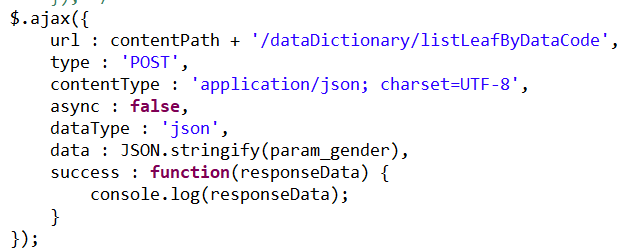




2.415错误







# 运行环境搭建常见问题

## 1.如何将项目运行环境转化为MAVEN内置的tomcat运行环境？

在项目的pom.xml下寻找合适的位置加入相关的jar包和tomcat配置

<servlet-api.version>2.5</servlet-api.version>

<jsp-api.version>2.0</jsp-api.version>

<!-- servlet相关 -->

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>servlet-api</artifactId>

<version>${servlet-api.version}</version>

<scope>provided</scope>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jsp-api</artifactId>

<version>${jsp-api.version}</version>

<scope>provided</scope>

</dependency>

此外需要配置tomcat的内容和资源文件路径

<build>

<finalName>ECM</finalName>

<plugins>

<!-- 资源文件拷贝插件 -->

<plugin>

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

<artifactId>maven-resources-plugin</artifactId>

<version>2.7</version>

<configuration>

<encoding>UTF-8</encoding>

</configuration>

</plugin>

<plugin>

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

<artifactId>maven-compiler-plugin</artifactId>

<version>2.3.2</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

<encoding>UTF-8</encoding>

</configuration>

</plugin>

</plugins>

<pluginManagement>

<plugins>

<!-- 配置Tomcat插件 -->

<plugin>

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

<artifactId>tomcat7-maven-plugin</artifactId>

<version>2.2</version>

<configuration>

<port>8080</port>

<path>/</path>

</configuration>

</plugin>

</plugins>

</pluginManagement>

<resources>

<resource>

<directory>src/main/java</directory>

<includes>

<include>\*\*/\*.properties</include>

<include>\*\*/\*.xml</include>

</includes>

<filtering>false</filtering>

</resource>

<resource>

<directory>src/main/resources</directory>

<includes>

<include>\*\*/\*.properties</include>

<include>\*\*/\*.xml</include>

</includes>

<filtering>false</filtering>

</resource>

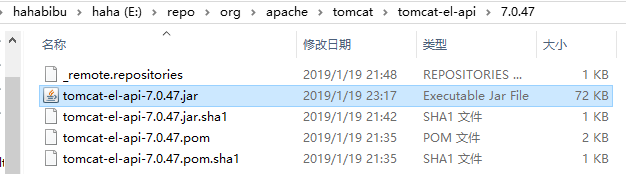
</resources>

</build>

可能出现的问题;

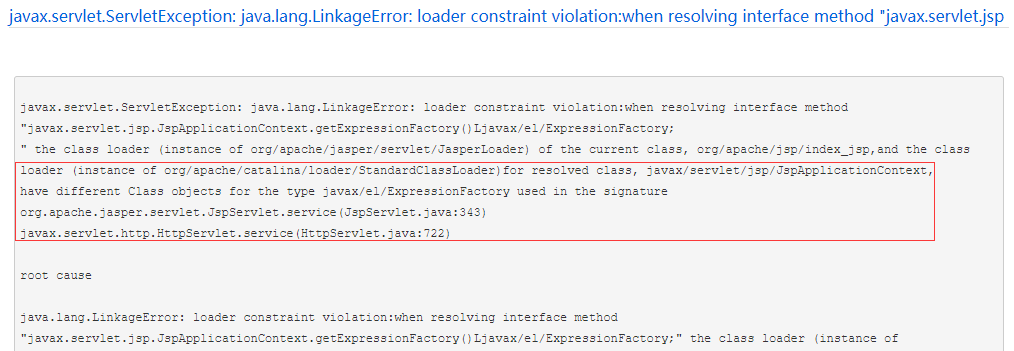
**maven tomcat7启动,访问资源时出现java.lang.ClassNotFoundException: javax.el.ELManager**

可能是由于tomcat7提供的el版本过低导致出错（tomcat7提供的el-api.jar版本过低，没有ELManager这个类），如果是外置tomcat则只需要更换tomcat8即可，此处也可换为maven的tomcat8，但中央仓库并没有这个tomcat8插件，从而有可能导致资源文件无法下载引起tomcat启动失败的问题，可以参考网上相关的资源配置文件将maven的仓库位置进行修改，或者是替换tomcat7下的tomcat-el-api.jar包。（在Maven Repository搜索 [javax.el](https://mvnrepository.com/artifact/javax.el)下载javax.el-api.jar3.0以上的版本，随后找到自身配置的仓库位置下org.apache.tomcat，寻找相关的文件进行替换，只需要替换jar包，修改为对应原来jar的名称即可）



Tomcat启动出现无法找到相关的资源配置文件

此处不考虑在pom.xml文件中直接引入，如果在pom.xml文件中直接加载el-api依赖则在运行的时候会报错，是由于在pom.xml文件中引入的jar不能够和运行环境即tomcat加载的相关jar包有所冲突，会导致访问资源的时候加载类冲突，从而报500错误



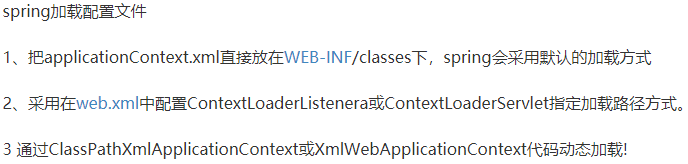
**Tomcat启动，出现资源配置文件无法找到的问题**

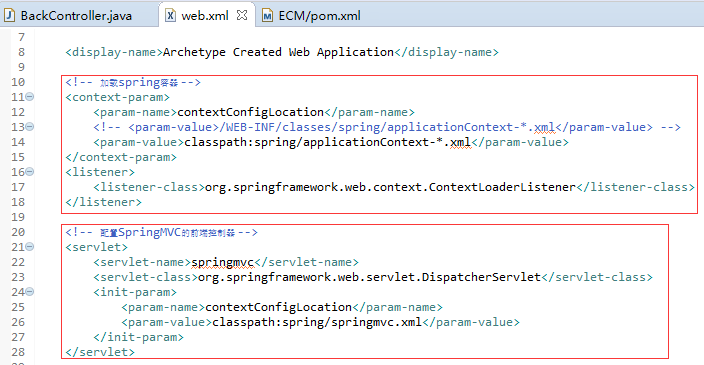
**Could not open ServletContext resource [/WEB-INF/applicationContext-\*.xml]**

**BeanFactory not initialized or already closed - call ‘refresh’ before accessing beans via the ApplicationContext**

问题分析：问题主要由于加载spring的默认配置文件位置一般是在/WEB-INF/下找applicationContext.xml文件。而此处eclipse中SSM项目所用到的applicationContext相关的文件是放在/java/src/resource/applicationContext-\*.xml。因此在默认加载中是找不到相关的applicationContext-\*.xml文件的，如果是引用外部tomcat的配置则不需要配置，也能够执行。此处使用内置的tomcat，tomcat启动时无法找到相关的配置文件，导致报错！

解决方案：需要在pom.xml中引入相关资源文件的配置，此外在web.xml中需要指定applicationContext-\*.xml的位置，修改如下：



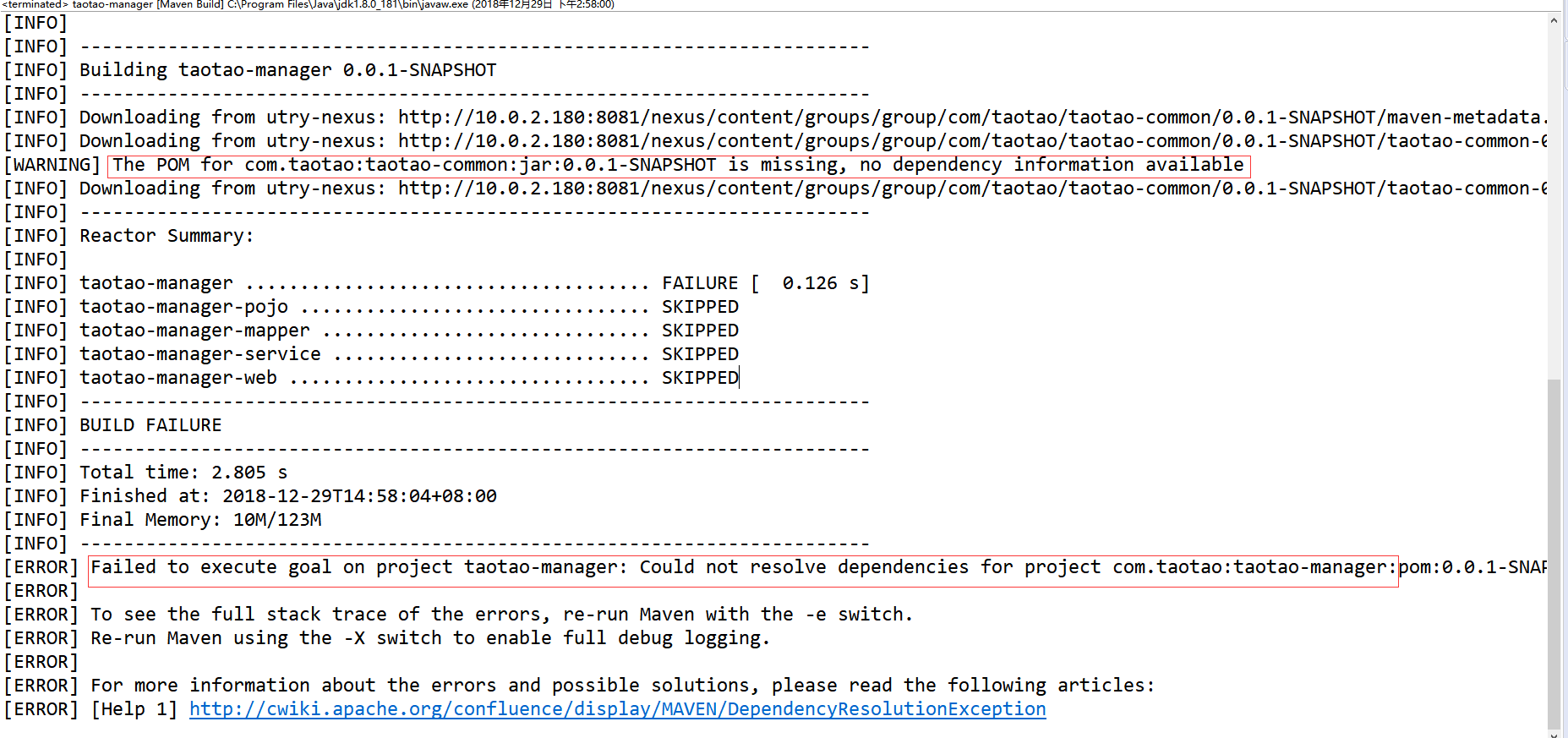




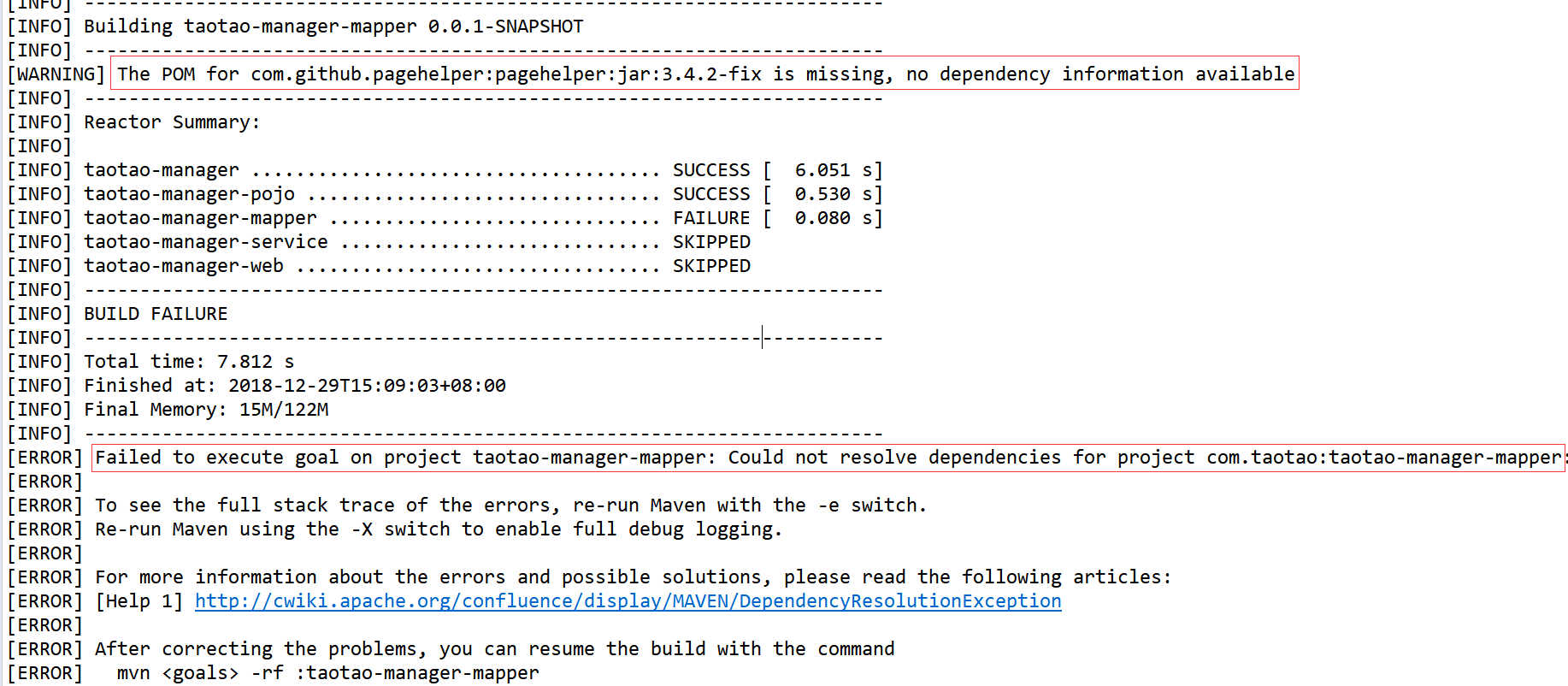
## 2.构建Maven工程时遇到的问题

（以下是构建taotao商城时遇到的问题和相应的解决方案）

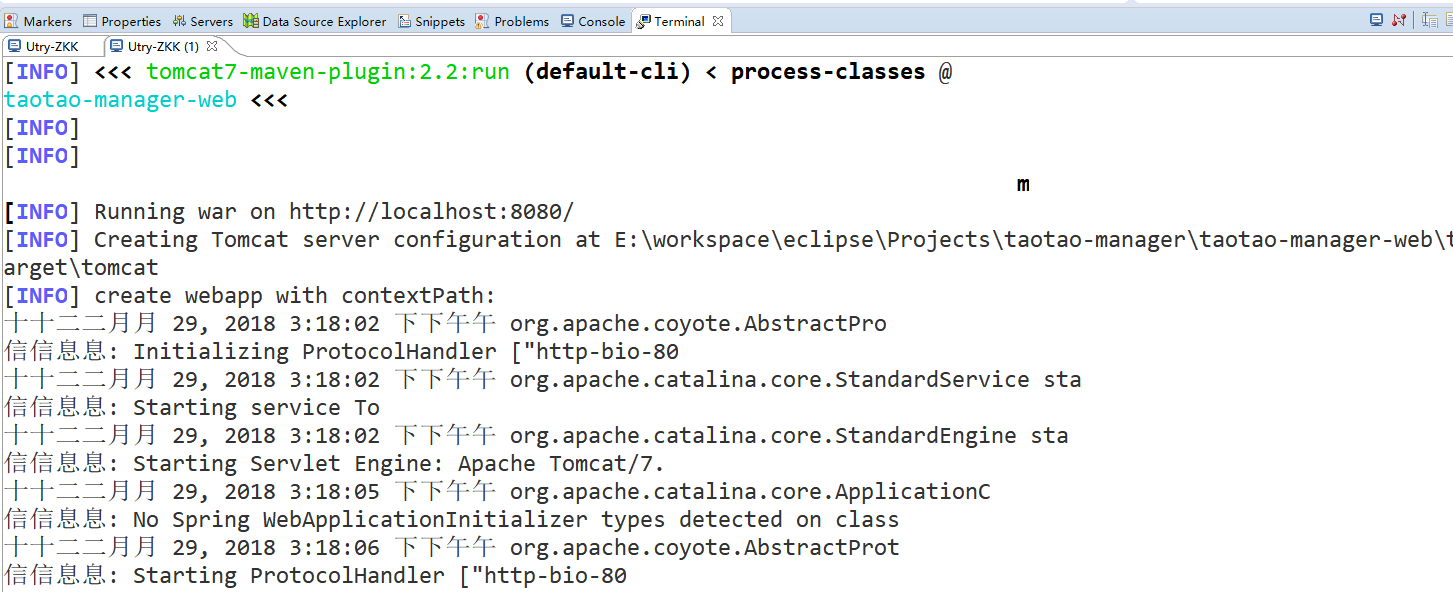
执行maven相关的指令，如果运行错误，则查看相应日志文件：(考虑可能是相关工程没有导入本地仓库)



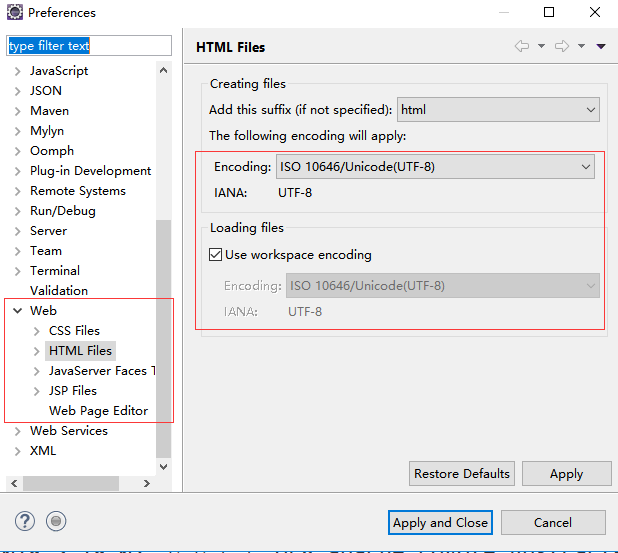
下述问题是由于mvn引入的jar包失效引起，由于taotao-parent中虽然修改了之前报错的插件，但没有重新将taotao-parent打包到本地仓库，实际上运行时工程访问的是本地仓库打包的内容，此处要注意这一点，每次修改工程，必须重新install，从而避免“信息不同步”导致的问题（**建议不要修改插件版本，会导致后续分页出错，因为使用的pagehelper插件引入的是修改后的fix版本，把相关的版本导入即可，如果更改成更高的版本，处理方式不同**）



通过终端启动，启动成功显示下列信息



通过网页访问：localhost:8080/,如果出现乱码问题，则可能是jsp编码格式（默认系统设置则需要修改成UTF-8）



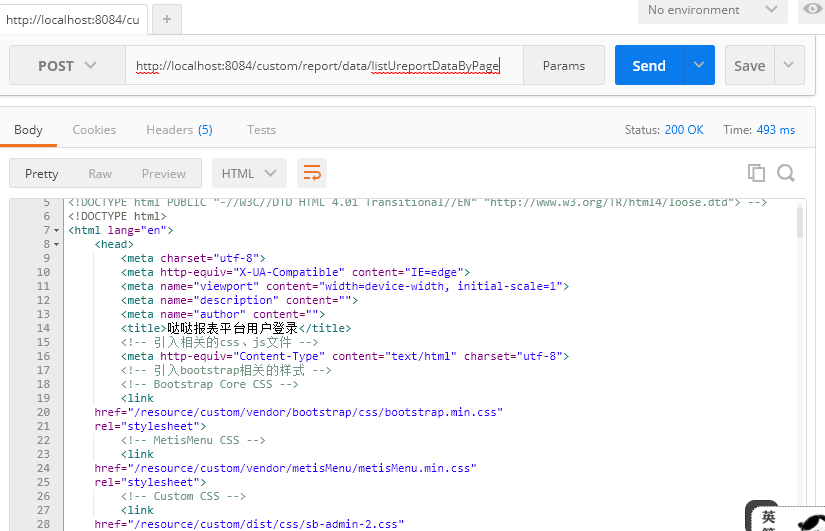
跨域调用服务dada-report请求的时候：

出错排查原因

1. 访问路径是否正确（404）
2. 请求方式是否正确（405、500）
3. 请求参数是否正确（405、500）
4. 服务层是否设置了拦截器（针对无参数返回的情况）
5. 内部服务器调用问题则判断代码逻辑设计上是否出现了问题

没有报错，默认跨域请求成功，却没有返回数据

考虑是不是在dada-report设置了拦截器，实际上请求是被拦截器拦截，并返回相应的请求数据，但返回的数据并不是自己指定的数据，从而导致出错（但实际上代码逻辑并没有问题）



不要在服务层设置拦截器！！！

如果要进行拦截则可以通过ThreadLocal保存用户登录信息，从而判断拦截对象

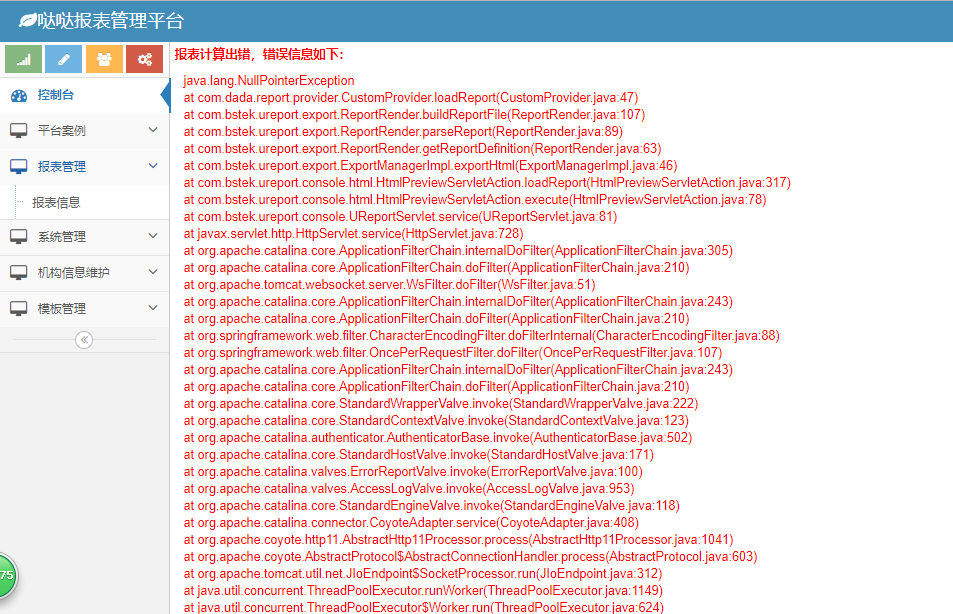
两种处理方式

直接传入

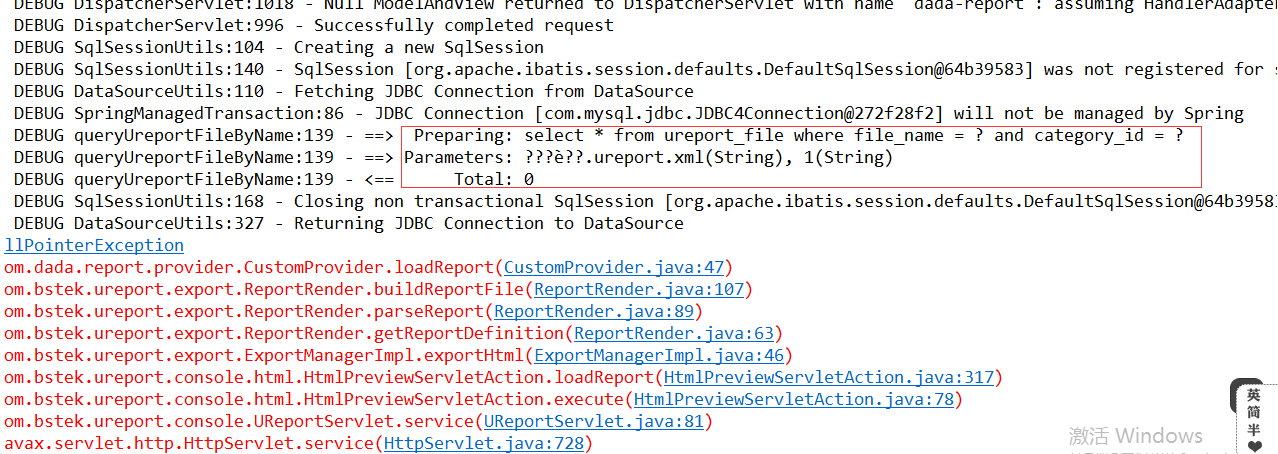
DataManagerDTO参数

直接传入page参数，再封装为DataManagerDTO，随后调用相应的服务

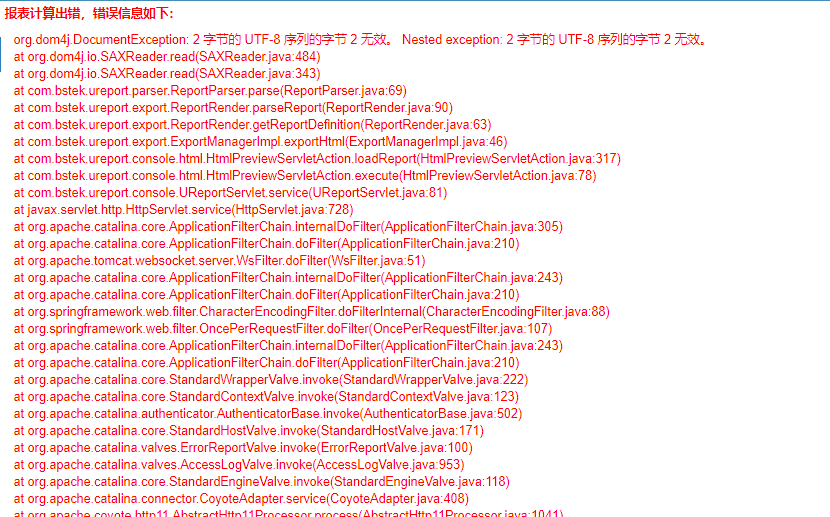
从浏览器中传入参数乱码问题（在预览报表的时候，如果报表名称含有中文，通过url传入的参数出现乱码问题，导致报表数据查找出错，从而引发空指针异常，报表设计器无法正常加载报表信息实际页面显示如下错误）



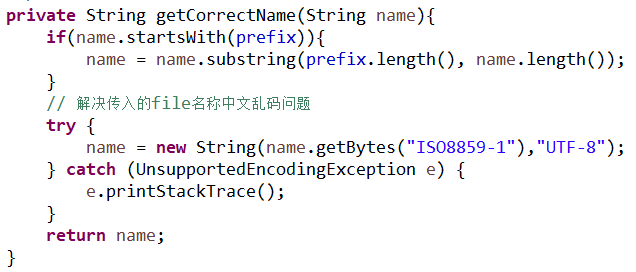
解决参考链接：https://blog.csdn.net/abcd1f2/article/details/42180969



如果处理空指针异常：（不能直接将string字符串转化为byte[]数据类型，数据读取之后报如下错误）

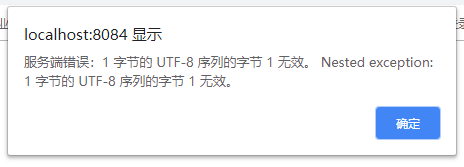


解决中文乱码，将传入的fileName的编码进行处理（修改了之后，报表数据存储又出现问题，存储的是乱码，也无法正常打开报表数据），

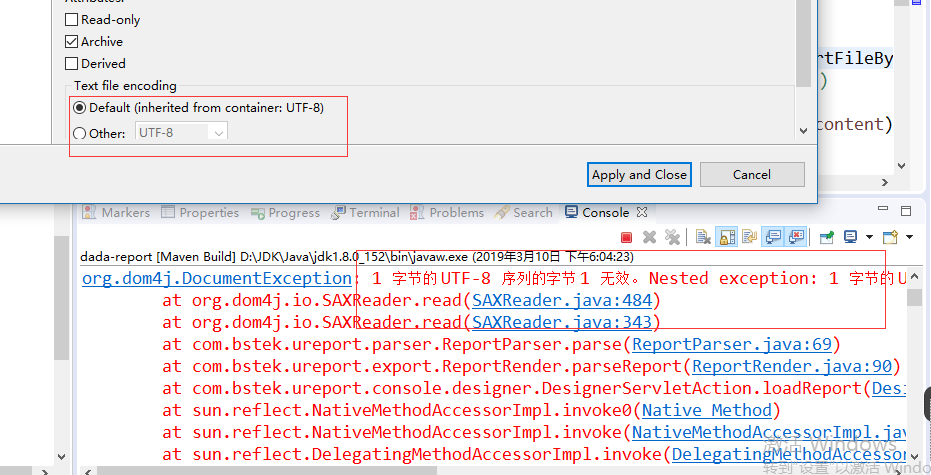


不在getCorrectName处修改，只针对加载报表信息方法的数据进行处理（虽然能够正常查找数据，但返回报表数据的时候仍然抛出异常信息）





出现上述问题的原因在于误修改了数据，导致字节序列无效（修改完成之后，需要重启服务器，删除之前编辑的脏数据，清空浏览器默认缓存，重新测试数据）



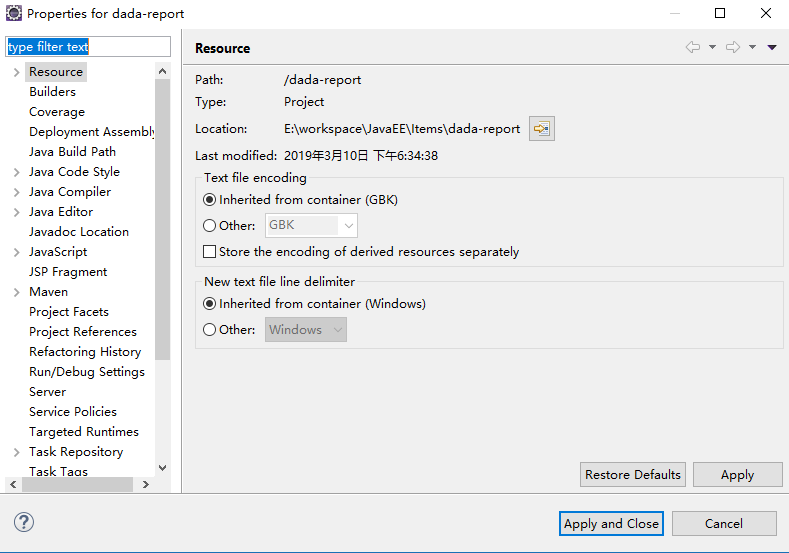
更换环境再运行dada-report时，发现除却存储的旧数据能够正常访问之外，新增的数据失踪无法正常访问

数据存储后不能正常获取，之前存储的数据能够正常查阅。新增的数据却不能够正常访问 。。

考虑是本地运行环境配置问题：发现所有工程编码默认为GBK

（关于eclipse编码设置问题参考链接：修改相关的配置，清除脏数据，再次进行测试

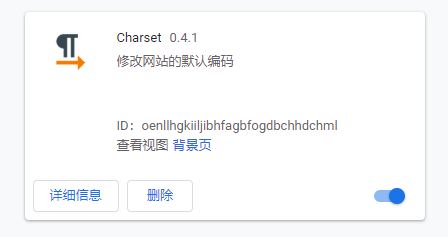
https://blog.csdn.net/lanmuhhh2015/article/details/79366872）



既然不能在代码中写死编码格式，唯有设置浏览器编码属性

要么在页面中设置编码属性

Chrome设置页面编码属性：下载charset插件，将解压后的“Charset\_v0.4.1.crx”拖到chrome浏览器中，等待弹框出现选择“添加至扩展程序”，添加完成之后可在扩展程序页面中查看插件加载是否成功，插件安装成功后，在浏览器页面任意位置点击右键，选择Charset则可根据相应的需求完成编码格式设置



隐藏bug：由于报表的共有性，当用户在浏览报表数据的时候，如果另一个用户将报表数据删除，此时用户如果不刷新页面，直接点击预览的时候访问的是缓存中的数据，此时是能够直接访问到报表信息而不报错，但是如果用户是 先清理浏览器缓存之后再点击预览访问报表数据，此时ureport缓存中是没有该报表信息存在的，需要再次从数据库中获取，此时再次访问数据库时拿到的数据是不存在的，因此直接调用该报表对象获取相应的fileContent属性会报空指针异常，需要在代码中处理空指针异常（没有办法处理则需要指出该问题）



在dada-portal直接想要获取相应的数据信息，出错，ureport并没有在dada-report中配置，因此不能直接获取，可尝试通过controller执行操作

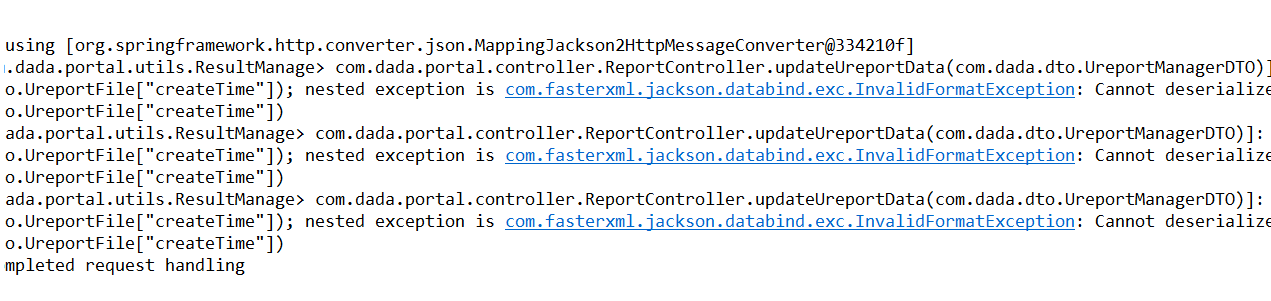


在dada-report中这和好数据，提供页面接口供其获取

针对模板展示，链接直接链接到dada-report的数据中

待解决问题：报表名称中文乱码问题？？？无法直接在代码中修改，需要

400请求问题，数据被拦截（项目代码问题：报错），不调用接口，排查代码问题



查找错误，并非数据被拦截，而是日期转换出现异常，需要在实体类上加注解

Ztree数据：

参考API文档、属性定义要相对应

如果出现数据没有正常显示或者是显示的视图全部为子节点

则考虑屏蔽某些不需要自定义的 属性，以免这些属性对数据的显示造成影响

针对框架嵌合出现

window.location.href的target控制

在使用框架时，经常会对框架子页面进行页面引导的情况，如果只是简单的设置location. href="",会使得整个页面显示在子框架中，导致出现重叠框架。如何解决呢？很简单。只要指定子框架的父级框架页面链接至目标页面即可。如

window.top.framename.location.href = url;

window.parent.location.href=url;

mainframe.location.href=url;

## 调用dada-rest服务（跨域请求处理）：

### 请求方式分析

Jsp页面能跨服务器请求页面信息，却无法跨域获取数据请求

此处实际场景分析如下：

由dada-rest提供基本的服务（访问数据库实现基本的CRUD），供dada-portal封装调用完成相关的数据封装和处理等操作

借助HttpClientUtil工具类实现跨域请求其他服务器的内容，主要有两种请求方式

doGet方式：（执行返回的结果为json字符串，根据需求解析字符串再进行处理，此种方式常常用于查找、访问数据），例子如下所示

String json = HttpClientUtil.doGet(REST\_BASE\_URL + REST\_ROLE\_INFO\_ADD);

doPostJson方式：（此种请求方式用于带有json参数的数据，借助JsonUtils工具类将经过处理后的Object对象转化为json字符串，再通过相应的方法调用将json字符串传给相应的服务器接口，调用接口服务完成数据访问），例子如下所示

String json = JsonUtils.objectToJson(object);

String jsonResult = HttpClientUtil.doPostJson(URL, json);

#### 1.Controller直接请求rest参考

@Controller

@RequestMapping("/portal/system")

**public** **class** SystemController **extends** BaseController{

@Value("${REST\_BASE\_SYSTEM\_URL}")

**private** String REST\_BASE\_SYSTEM\_URL;

@Value("${REST\_ROLE\_INFO\_ADD}")

**private** String REST\_ROLE\_INFO\_ADD;

@Autowired

**private** CommonService commonService;

// 添加角色信息

@RequestMapping(value = "/addRoleInfo", method = RequestMethod.***POST***, consumes = "application/json;charset=utf-8")

@ResponseBody

**public** String addRoleInfo(@RequestBody RoleInfo roleInfo,HttpServletRequest request, HttpServletResponse response) {

roleInfo.setRoleId(IDUtils.*genRandomUUId*());

UserInfo loginUser = commonService.getUserInfoByToken(request, response);

**if**(loginUser!=**null**) {

// 设置默认属性(当前用户所属机构id)

roleInfo.setCategoryId(loginUser.getCategoryId());

}**else** {

// 当前用户不存在,机构信息设置默认机构(-1)

roleInfo.setCategoryId(ConstantUtils.***DEFAULT\_DATA***);

}

// 调用rest服务:添加角色信息

// 1.将要传送的数据(对象)转化为json字符串

String json = JsonUtils.*objectToJson*(roleInfo);

String jsonParam = JSON.*toJSONString*(roleInfo) ;

// 2.通过HttpClientUtil.doGet(...)方法请求rest服务器;

// String jsonResult = HttpClientUtil.doPostJson(REST\_BASE\_SYSTEM\_URL + REST\_ROLE\_INFO\_ADD,jsonParam);

String jsonResult2= HttpClientUtil.*doPostJson*(REST\_BASE\_SYSTEM\_URL + REST\_ROLE\_INFO\_ADD,json);

/\*

\* ResultManage result = JSON.parseObject(json, ResultManage.class);

\* Map<String, Object> resultMap = result.getData();

\* UserInfo userInfo = JSON.toJavaObject((JSONObject)resultMap.get("loginUser"),UserInfo.class);

\*/

// return getJsonResult(resultMap);

**return** jsonResult2;

}

}

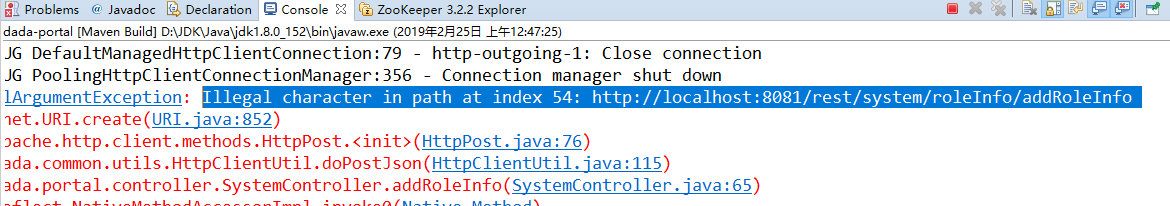
#### 2.封装service调用rest服务参考

依次封装service、controller层，在service层调用相应的rest服务

### 测试过程中遇到的问题和解决方案

**Illegal character in path ... 非法字符......**

调用服务的时候报错“非法的字符请求”，原因是在配置rest路径的时候在路径后方不小心添加了空格，导致系统通过HttpClientUtil请求rest服务的时候无法识别访问该路径，导致空指针异常，因此在编写配置文件的时候要细心检查文件，避免微小问题!



问题：跨域请求返回数据报406错误：

参考链接：https://blog.csdn.net/happyljw/article/details/78350877

FastJSON、JsonUtils相应转化数据格式如下：

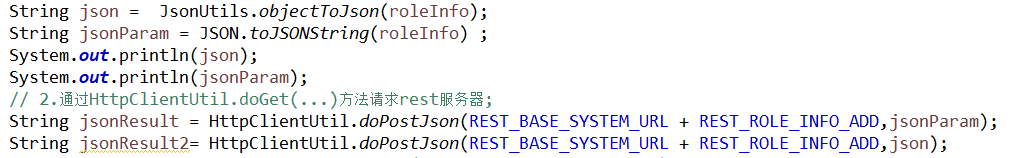
String json = JsonUtils.*objectToJson*(roleInfo);

String jsonParam = JSON.*toJSONString*(roleInfo) ;

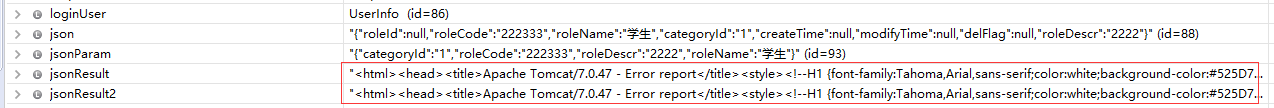
// 通过HttpClientUtil.doGet(...)方法请求rest服务器;

String jsonResult = HttpClientUtil.*doPostJson*(REST\_BASE\_SYSTEM\_URL + REST\_ROLE\_INFO\_ADD,jsonParam);

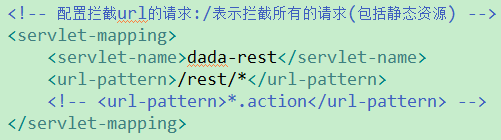
String jsonResult2= HttpClientUtil.*doPostJson*(REST\_BASE\_SYSTEM\_URL + REST\_ROLE\_INFO\_ADD,json);



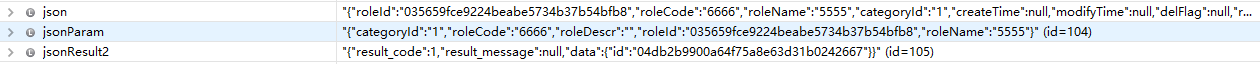
Debug测试结果显示，由JsonUtils工具类转化的json字符串为一个实体的数据（封装为json对象），会相应将没有数值的参数值设置为null；而通过JSON.toJSONString方法只是将前端页面传递的参数转化为相应的json字符串（只封装已有的属性）



测试请求rest服务的时候报错，需要参考出错的日志信息，随后根据相应的内容进行调整：例如404错误则是访问路径出错，需要参考相应Controller设置的url，或者查看web.xml中拦截配置的路径是否有误，此处404访问路径错误，是由于dada-rest中web.xml配置路径拦截多加了一层目录导致访问路径出错（修改为默认拦截所有路径）



相应的其他错误均可以通过debug调试，如果测试成功则显示如下信息，相应返回数据结果



### 服务调用需要注意的问题：

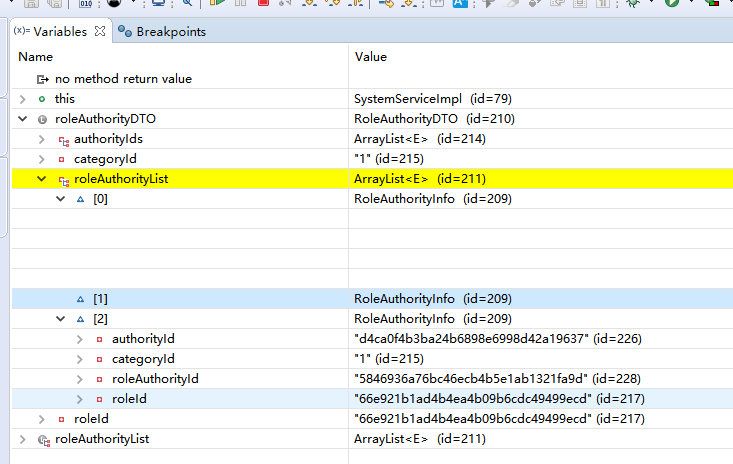
调用rest服务可能会遇到500、404等球球服务错误，需要设置断点一一进行排除出错点，常见的404错误访问路径问题，500请求服务器出错可能是请求方式、请求参数的问题，此处分页查找根据与前端嵌合的方式进行不同的处理

方式1：返回的分页数据：PageData，请求分页参数Page

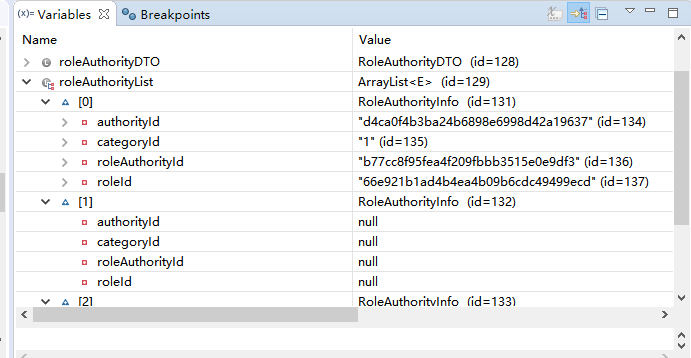
方式2：利用dataTable实现数据自动分页，其对返回的数据参数有一定的要求

### List类型数据处理问题

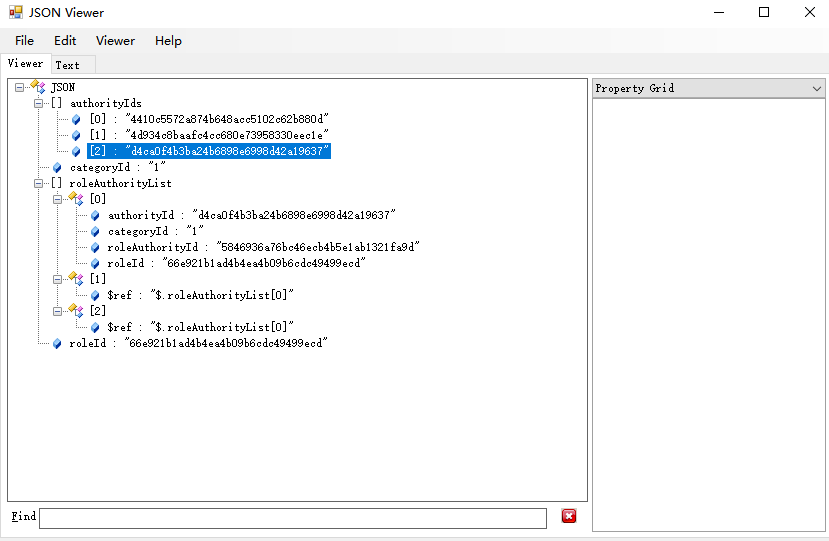
在处理用户权限信息时，希望通过用户传递authorityIds列表数据，由portal接口接受数据封装成相应RoleAuthorityInfo数据，封装的列表传送到后台报错，仔细查看封装后的数据显示如下内容，加载角色权限数据：



dada-rest中接收到的RoleAuthority数据显示如下：



说明前台接收到的数据经过封装后并没有正常进行处理，查看通过JSON转化后的数据，显示如下内容，说明代码逻辑处理部分出错，导致数据没有封装成对象并加载到相应的列表中

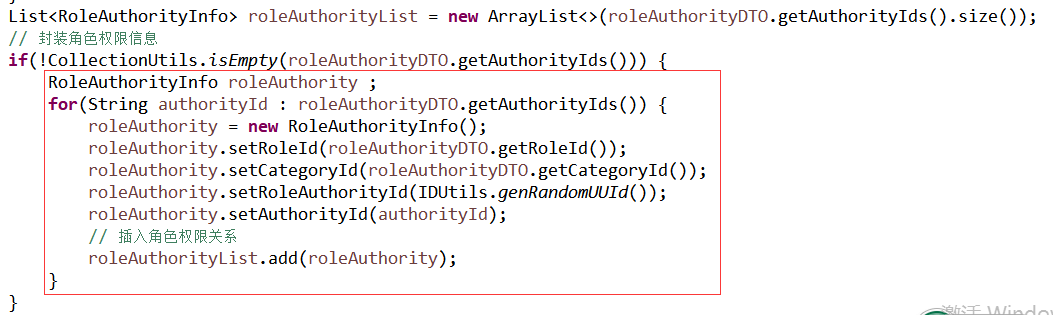


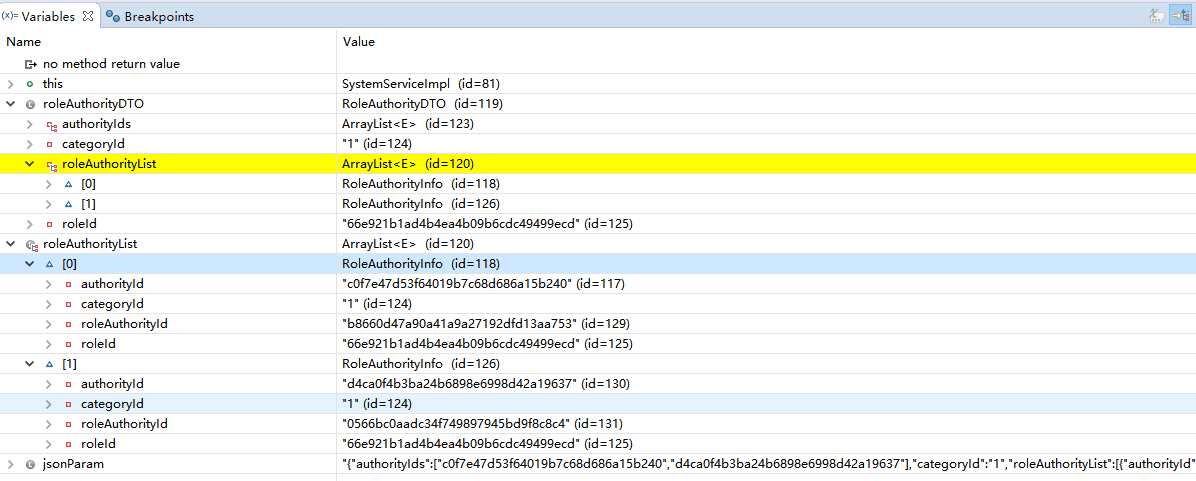
解决方案：

处理前：



处理后：原因分析：通过JSON处理数据，将RoleAuthorityInfo对象转化为相应的json数据，前者做法是只创建了一个对象，然后通过修改该对象信息完成数据填充，由结果显示，后面添加的数据都是第一个添加的数据的引用，因此在转化为JSON数据的时候就会出现上述错误（转化后的json数据并不是正常的JSON对象字符串），解决方案是使roleAuthorityList中的每个对象均指向不同的引用，在转化的时候方能正常转化





### 数据库连接报错

**[java.sql.SQLException: Lock wait timeout exceeded; try restarting transaction](https://www.cnblogs.com/zs-notes/p/9076694.html)**

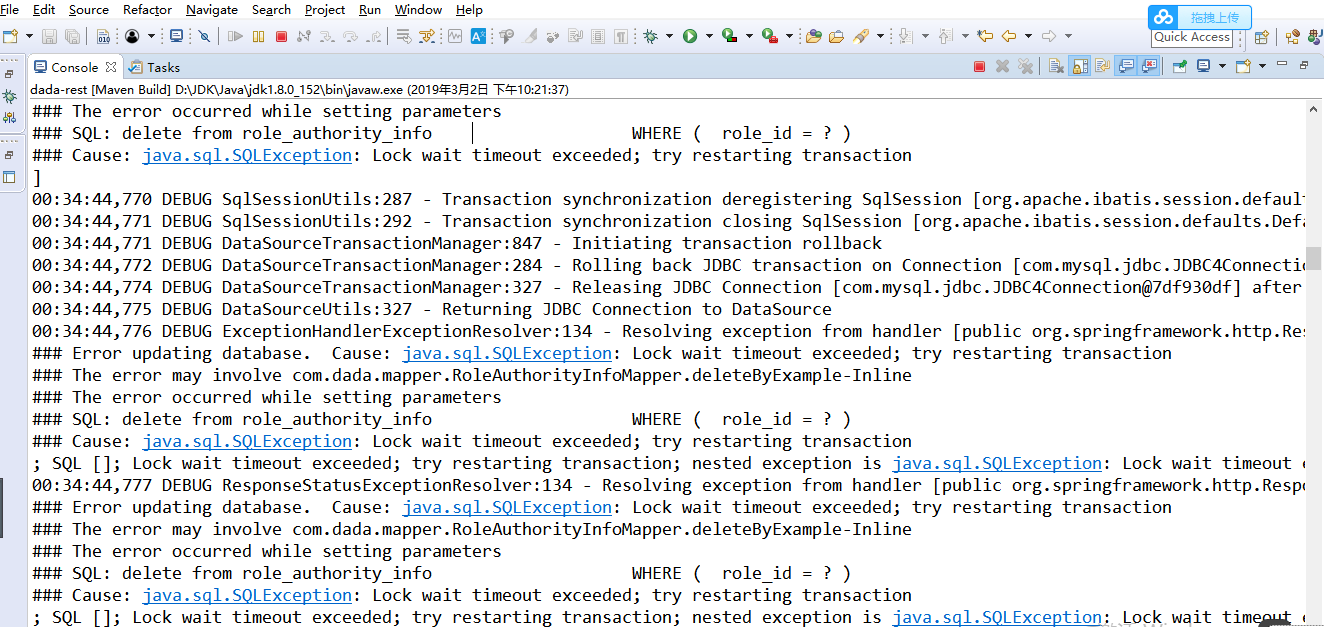
测试出现问题：

Error updating database. Cause: java.sql.SQLException: Lock wait timeout exceeded; try restarting transaction

### The error may involve com.dada.mapper.RoleAuthorityInfoMapper.deleteByExample-Inline

### The error occurred while setting parametersSQL: delete from role\_authority\_info WHERE ( role\_id = ? )

### Cause: java.sql.SQLException: Lock wait timeout exceeded; try restarting transaction]



参考链接：

<https://www.cnblogs.com/zs-notes/p/9076694.html>

考虑是测试的时候访问数据库连接过于频繁导致出错，重启项目，重启数据库连接，重新进行测试即可，但不能够止步于解决问题，需要分析问题来源！可以通过参考相关的链接分析真正出错的原因！

### 日期转化问题

在跳转修改数据页面的时候使用ajax封装页面信息，初始化的时候使用fmt转化日期数据，由于一开始的值为空字符串，所以在初始化转化的时候报类型转化错误。

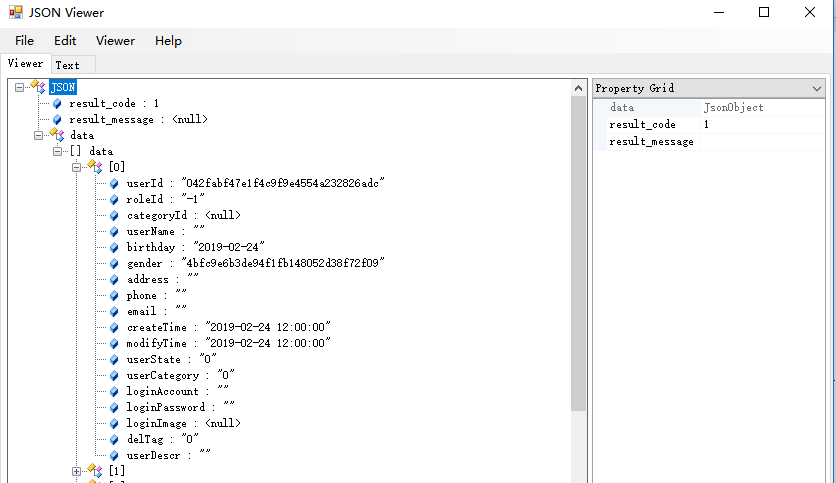
fmt:formatDate标签接收的value数据是一个Date类型，此处不能直接将字符串传入，可以在js中处理完数据之后再赋值，亦可不使用该标签，直接传入转化后的字符串数据



修改页面信息：



代码复制的时候需要注意路径访问问题，如果获取的参数转化成另一个对象，从代码上来说没有异常，但在执行的时候转化生成的对象数据始终为空，例如在获取category列表的时候不小心路径编写成调用userInfo列表，获取到的数据是UserInfo类型，而通过JSONArray转化成Category类型时，是无法识别到对应的属性，因此属性默认都封装为null，从而在调用属性的时候导致“空指针异常”，这种隐藏的问题难以一时察觉，需要时刻警戒，学会设置断点分析数据



## dada-sso单点登录系统调用

请求页面跳转前如何获取到当前登录的用户信息？



但是拦截器的调用必须是调用接口的时候方能够生效，如果通过配置过滤器可在不同的代码层获取到当前登录的用户信息，亦可借助ThreadLocal实现。

Json数据转换异常问题：



日期注解问题：

**import** com.fasterxml.jackson.annotation.JsonFormat;

在实体类上添加注解：

@JsonFormat(pattern = "yyyy-MM-dd hh:mm:ss", timezone = "GMT+8")

fastjson相关jar包引入：

<dependency>    
    <groupId>net.sf.json-lib</groupId>    
    <artifactId>json-lib</artifactId>    
    <version>2.4</version>    
    <classifier>jdk15</classifier>    
</dependency>

<dependency>

<groupId>com.alibaba</groupId>

<artifactId>fastjson</artifactId>

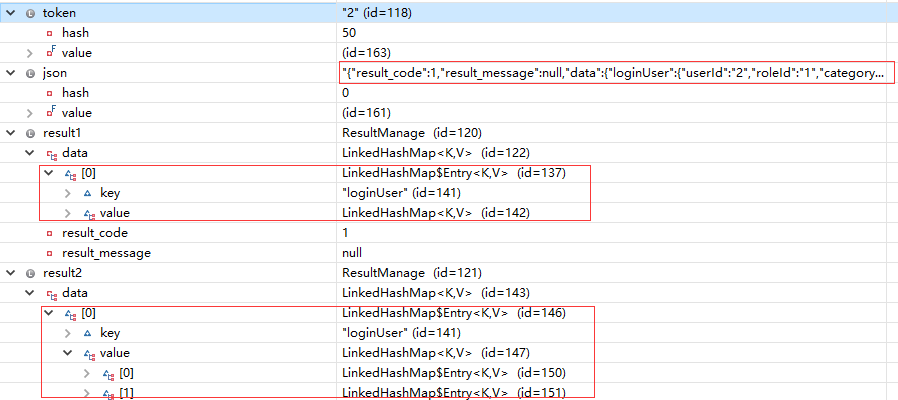
<version>1.2.56</version>

</dependency>

ResultManage result1 = ResultManage.*format*(json);

ResultManage result2= JsonUtils.*jsonToPojo*(json, ResultManage.**class**);

ResultManage result3 = JSON.*parseObject*(json, ResultManage.**class**);

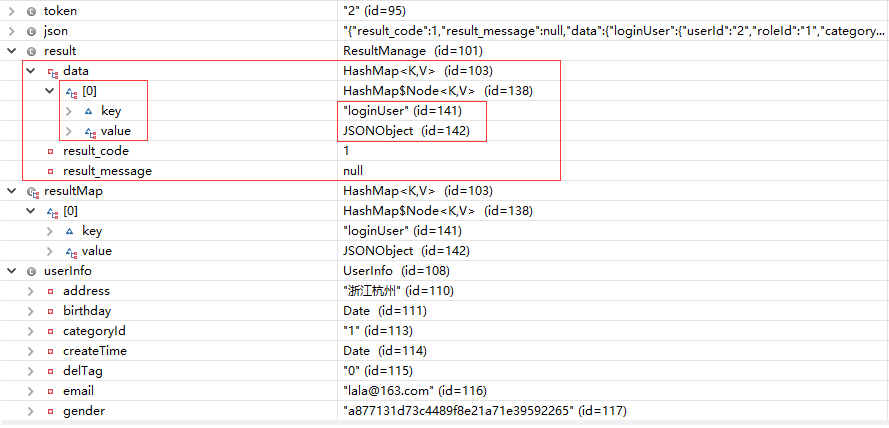


ResultManage result = JSON.*parseObject*(json, ResultManage.**class**);

Map<String, Object> resultMap = result.getData();

UserInfo userInfo = JSON.*toJavaObject*((JSONObject)resultMap.get("loginUser"),UserInfo.**class**);

使用fastjson处理数据，与jackjson、自定义的JsonUtils处理的数据返回结果不同，因此在转化的时候报类型转化错误，且由JSONObject转化为相应的实体时也要遵守相应的规则，否则包类型转化错误或其他问题（JSONObject类型为com.alibaba.fastjson.JSONObject;）



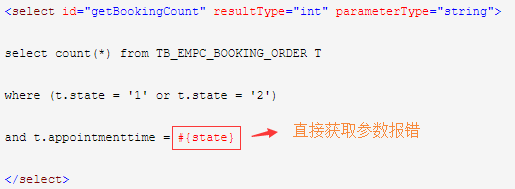
## 常见问题分析处理

### Mybatis文件异常处理

[Mybatis异常There is no getter for property named 'XXX' in 'class java.lang.String'](https://www.cnblogs.com/orac/p/6726323.html)，当传入当入参为 string类型时 (包括java.lang.String.)

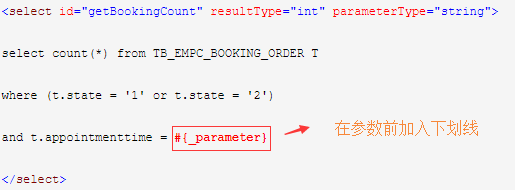
如果直接使用#{xxx}引入参数.会抛异常There is no getter for property named 'XXX' in 'class java.lang.String'

原因分析:Mybatis默认采用OGNL解析参数，所以会自动采用对象树的形式取 string.xxx 值，如果没有在方法中定义,则会抛异常报错。

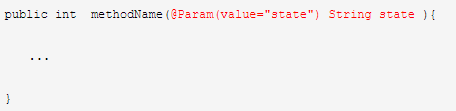


以下提供两种解决方法：

解决方案1：在对应mapper.xml文件中把#{xxx}修改为 #{\_parameter}



解决方案2：在指定方法参数前添加注解@Param()



### Jar包引入错误

**问题分析：**导入displaytag标签之后启动项目报错，原因在于displaytag标签导入需要依赖相关的jar包，可以通过maven仓库查找displaytag需要依赖的jar包，可能出现如下问题：

三月 01, 2019 3:37:47 下午 org.apache.catalina.core.ApplicationContext log

信息: No Spring WebApplicationInitializer types detected on classpath

三月 01, 2019 3:37:48 下午 org.apache.catalina.core.ApplicationContext log

信息: Initializing Spring root WebApplicationContext

三月 01, 2019 3:37:48 下午 org.apache.catalina.core.StandardContext listenerStart

严重: Exception sending context initialized event to listener instance of class org.springframework.web.context.ContextLoaderListener

java.lang.NoSuchMethodError: org.slf4j.spi.LocationAwareLogger.log(Lorg/slf4j/Marker;Ljava/lang/String;ILjava/lang/String;Ljava/lang/Throwable;)V

at org.apache.commons.logging.impl.SLF4JLocationAwareLog.info(SLF4JLocationAwareLog.java:120)

at org.springframework.web.context.ContextLoader.initWebApplicationContext(ContextLoader.java:285)

at org.springframework.web.context.ContextLoaderListener.contextInitialized(ContextLoaderListener.java:106)

at org.apache.catalina.core.StandardContext.listenerStart(StandardContext.java:4939)

at org.apache.catalina.core.StandardContext.startInternal(StandardContext.java:5434)

at org.apache.catalina.util.LifecycleBase.start(LifecycleBase.java:150)

at org.apache.catalina.core.ContainerBase$StartChild.call(ContainerBase.java:1559)

at org.apache.catalina.core.ContainerBase$StartChild.call(ContainerBase.java:1549)

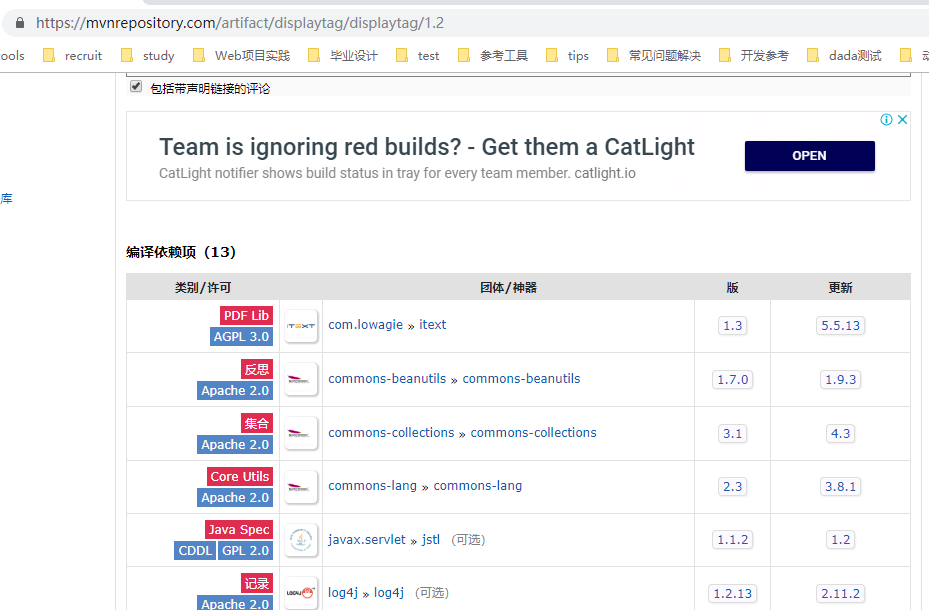
at java.util.concurrent.FutureTask.run(FutureTask.java:266)

at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)

at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)

at java.lang.Thread.run(Thread.java:748)

**解决方案：引入相关jar包**



此处参考如下依赖：

<dependency>

<groupId>displaytag</groupId>

<artifactId>displaytag</artifactId>

<version>......</version>

</dependency>

<dependency>

<groupId>commons-logging</groupId>

<artifactId>commons-logging</artifactId>

</dependency>

<dependency>

<groupId>log4j</groupId>

<artifactId>log4j</artifactId>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-log4j12</artifactId>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

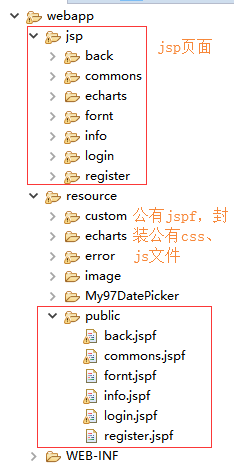
<artifactId>slf4j-api</artifactId>

</dependency>

### 有关前端页面整合路径问题

方式1：将要引用的resource文件（css、js、icon）和后台jsp直接放置在webapp目录下，浏览器访问可以直接通过localhost:端口到达webapp目录

根据相应路径访问文件信息，对应的jsp如果是直接放在webapp目录下，则浏览器发布时亦可直接访问，当这种形式并不符合实际应用要求（实际应用中需要将部分页面适当隐藏）



方式2：将要引用的文件信息（resource）文件置为内部访问，即在webapp下创建WEB-INF目录,在该WEB-INF级目录下创建相应的目录存储相关的resource、static、jsp文件，

在WEB-INF一级目录下的数据默认是私有的，是无法通过外部服务器访问的，如此一来，在实际执行的时候会出现jsp页面访问成功，而相关的样式、js却没有执行，是由于服务器无法直接

访问WEB-INF一级目录下的数据，因此需要在整合文件（例如springmvc.xml）下配置相应的资源映射，使得服务器能够对应到指定的目录

举例说明：

<mvc:resources location="/WEB-INF/resource/" mapping="/resource/\*\*"/> ：

分析：将/webapp/WEB-INF下resource级目录所有文件信息映射为/webapp/resource/\*\*

则服务器内部可直接通过：localhost:8080/resource/\*\*访问相关文件信息

对应：${pageContext.request.contextPath}/resource/\*\*

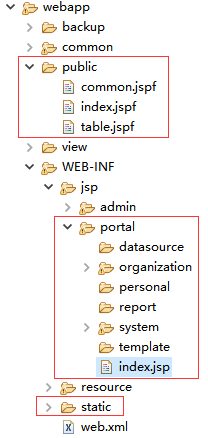
页面编译后显示为：举例说明：

<link rel="stylesheet" href="/resource/assets/css/bootstrap.min.css"/>

以此类推：分析<mvc:resources location="/WEB-INF/static/" mapping="/static/\*\*"/>

但对于一些公共的页面则需要放置在webapp级目录下，如果放在WEB-INF下还是无法直接访问，不过可以考虑通过接口实现页面跳转，为方便起见还是考虑将公有的数据（公有的jsp文件）放置在webapp下进行访问

此外，针对页面样式、js失效的问题，可以通过F12选择“Sources”选项卡依次查看错误信息，一一对应，可能是访问路径问题或是js内部引用、依赖的一些文件没有加载，都需要一一进行排查





### 浏览器渲染ajax动态拼接元素样式失效问题

 浏览器渲染ajax动态拼接元素问题:

[JS](https://www.baidu.com/s?wd=JS&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd" \t "https://blog.csdn.net/y666666y/article/details/_blank)执行完后会不过太久就渲染css（不等开出的异步js，只等主js完成）,所以如果使用ajax的话，数据加载过长时间，会导致css样式失效，初步考虑设置ajax的同步即可（async = false），包括${pageContext.request.contextPath}这种后台拼接过来的地址由于过了渲染阶段都不会识别这个内置对象会以为是一般的相对路径，所以<a>标签会直接加上上写文地址， 但是如果后台直接写成“http”那么浏览器会检测到这个http开头的是绝对路劲，可以直接访问，但缺陷在于导致后期开发调整维护工作量巨大

由于$.post() 和 $.get() 默认是异步请求，如果需要同步请求，则可在ajax执行前进行设置：

在$.post()前把ajax设置为同步：$.ajaxSettings.async = false;

在$.post()后把ajax改回为异步：$.ajaxSettings.async = true;

$.ajaxSettings.async = false;

$.post("/finance/getLastTimeCard", data, function(result) {

// 请求处理

},"json");

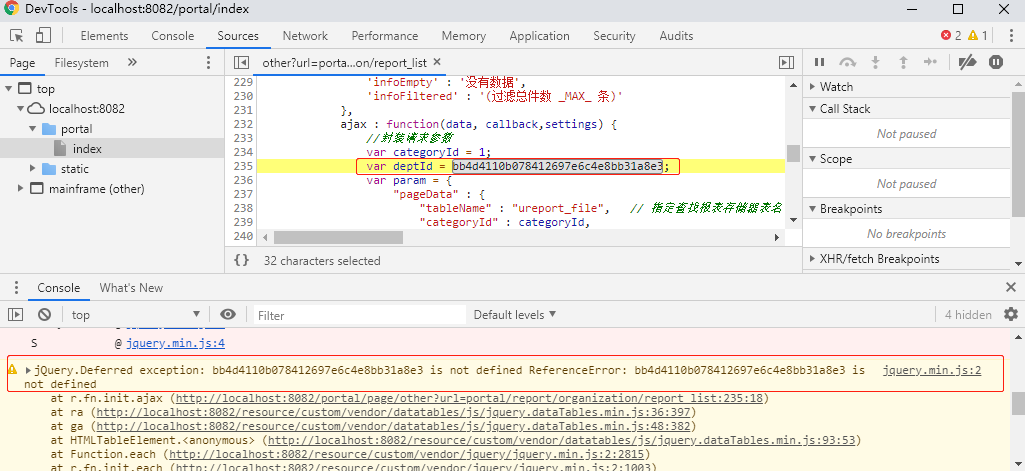
但使用bootstrap框架使用$.ajaxSettings.async = true;还是无效，参考网上的说明可以考虑重新调用api好点，同步网络慢会浏览器会假死，ajax放到ui初始化后执行同步也没用

为解决数据局部刷新与页面刷新(js、css)渲染的冲突,此处获取用户菜单信息不是通过ajax进行异步请求,即当页面js、css数据加载完成时,ajax请求却还未完成,此处通过ajax相应拼接或封装的html片段无法进行渲染,如果是EasyUI或其他的框架则可以调用相应的方法进行强制渲染,此处针对bootstrap框架还没有了解到有效的解决方案,可以考虑通过dom操纵节点,或是由后台分装数据转发到页面再进行处理

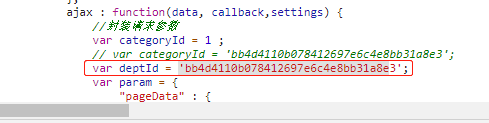
此处还存在一个ajax和c:foreach标签结合使用的误区，c:foreach是在页面加载的时候生成，通过ajax赋值给c:foreach是无法再次生成的（如果想使用ajax编辑某个表单值则可通过name或者id属性完成数据封装），此外可以在js里面动态生成html标签，随后嵌入heml页面中，但是这种方式操作需要重新渲染页面以防止js、css，还要考虑框架的嵌合。一句话总结：jstl的c:foreach遍历jsp域中的数据，而ajax获取的数据只能够通过js编写脚本，实现页面的动态效果

为了避免用户每请求一次页面就封装一次菜单信息,导致页面数据响应缓慢,此处不沿用后台管理框架,参考以往框架,在点击相应模块页面的时候,在当前页面中提供空白区域供子页面展示,除非用户手动刷新页面,否则不会请求菜单刷新借助后台通过model或request请求封装参数到前台页面中获取,再通过c:foreach标签遍历封装菜单信息(间接实现用户的权限管理)--此处暂不支持权限覆盖到按钮,仅仅针对某个模块进行权限验证、设置

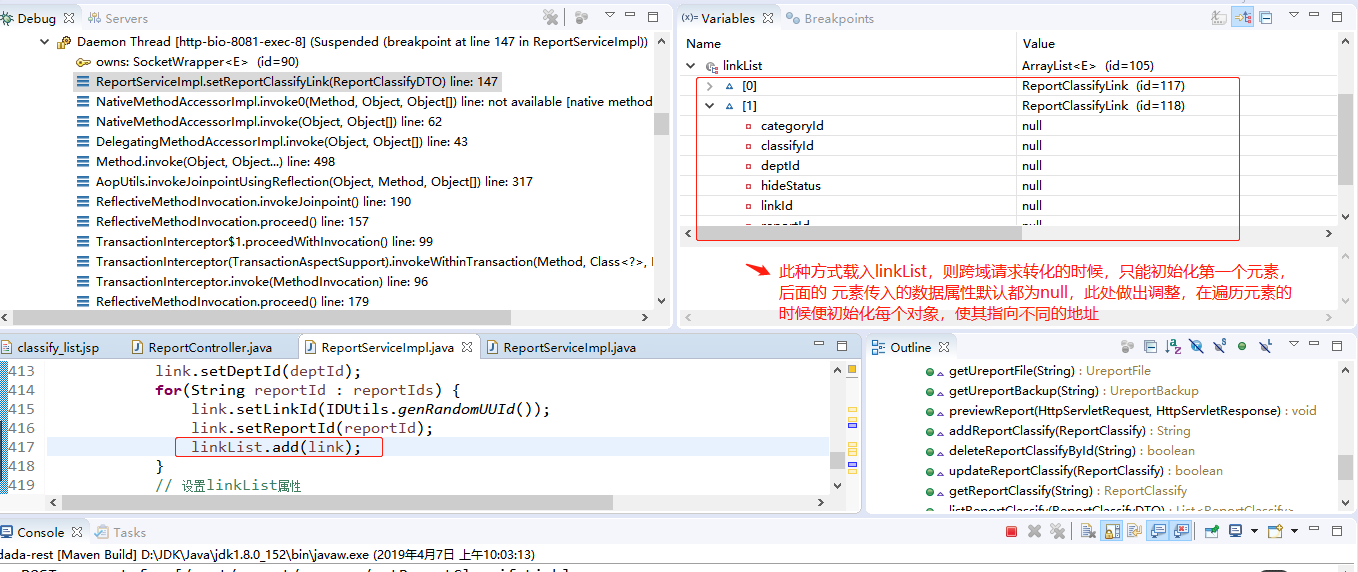
### JQuery字符串识别异常



将指定的内容调整为字符串形式即可



### LinkedList转化出错

****

## dataTable插件使用常见问题

### DataTable插件报错：

Uncaught TypeError: Cannot read property 'style' of undefined，

原因：table 中定义的列和Columns数据表的项数量对不上，需要仔细审查通过ajax获取的后台数据，然后进行封装

页面封装参考：

<https://blog.csdn.net/aojiancc2/article/details/56916187>

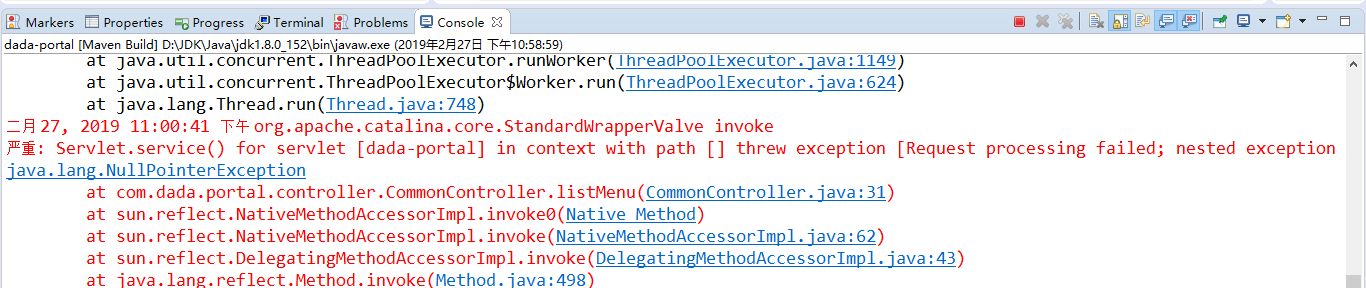
Json工具参考：

<http://www.cnblogs.com/vinn7/p/8110844.html>

## 调用Service服务空指针异常

严重: Servlet.service() for servlet [dada-portal] in context with path [] threw exception [Request processing failed; nested exception is java.lang.NullPointerException] with root cause

java.lang.NullPointerException：



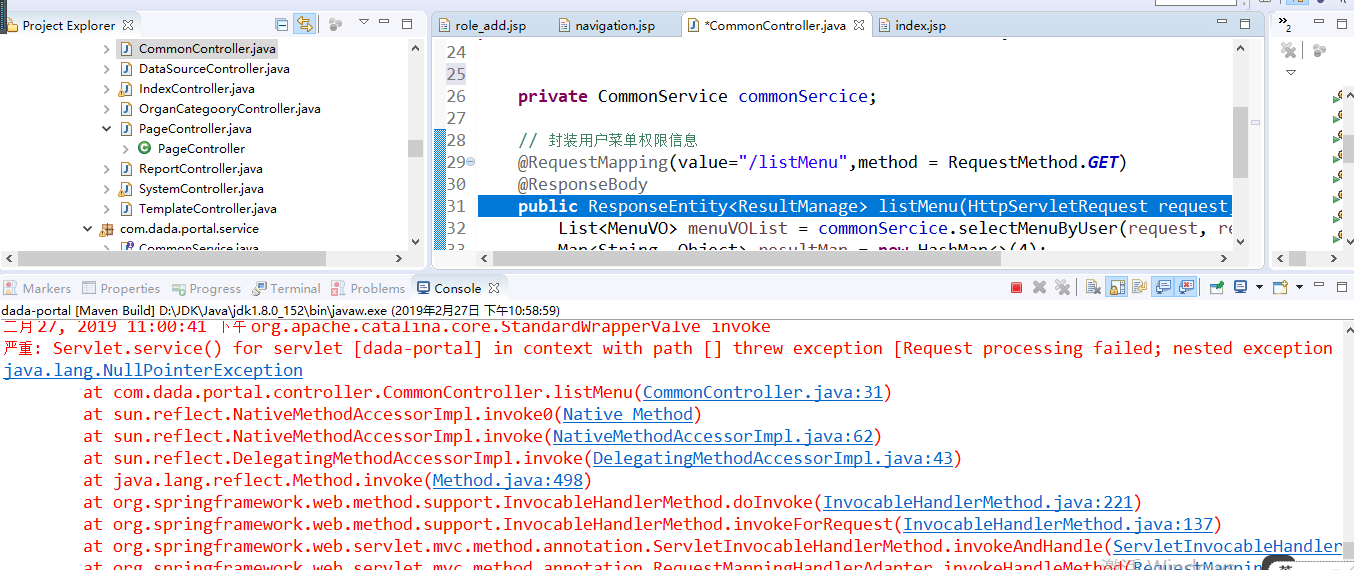
一开始考虑页面传过来的参数问题？？Jsp默认的九大对象相关

查找原因是相关的service没有初始化成功，没有加注解，代码逻辑上不报错，但是在系统运行初始化的时候报错

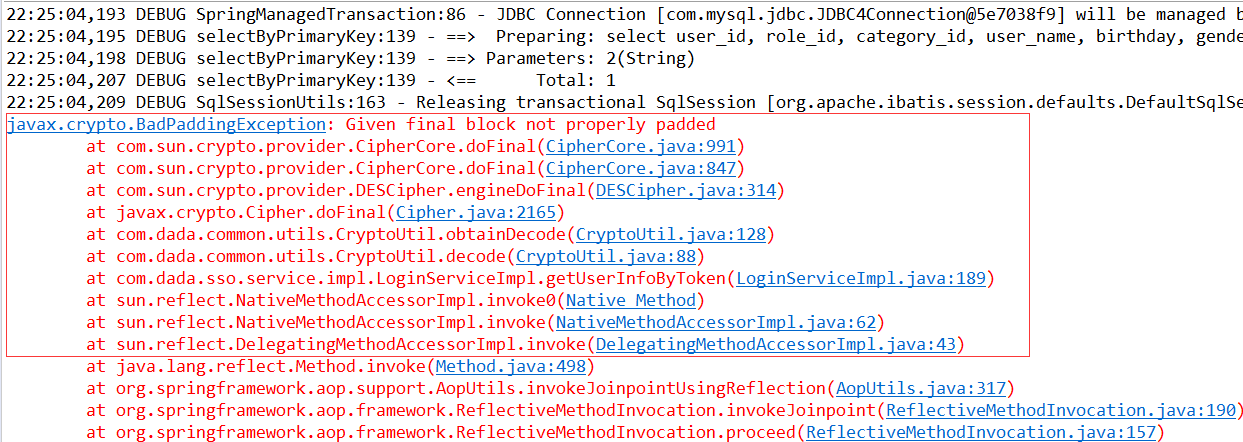
参考链接：

<https://blog.csdn.net/a447332241/article/details/75125305>

https://blog.csdn.net/thebigdipperbdx/article/details/80071800



## 密码加密解密失败



直接测试有效，但是整合到项目中报错（空指针异常）

秘钥不能随便更改，加密解密的秘钥要一致

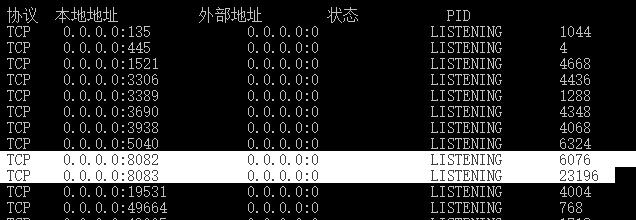
在测试的时候之所以出现这个问题是由于一些早先创建的脏数据的存在导致的

参考链接

<https://blog.csdn.net/rj042/article/details/8196125>

<https://www.cnblogs.com/digdeep/p/5580244.html>

## Maven内置tomcat插件启动报端口号冲突问题



一开始是考虑8082端口被其他项目启动所影响，但始终没检查出错误，而且此前项目也正常启动无误，后通过网上相关参考资料发现，是由于8082端口被某个应用程序占用，经过排查发现（在cmd查找占用该端口号的进程号，随后打开任务管理器，根据pid对应到当前启动的进程，发现8082端口被QQ应用程序占用，从而导致项目启动报错），因此之后再排查端口占用问题的时候要格外注意，不然在启动项目的时候默认关闭其他无用的程序，排查基本问题



## 多级分类构建

构建多级分类：

* 创建树结构
* 调用方法获取递归封装树形结构
* controller测试：获取所有数据，调用封装方法随后进行测试

参考链接：<https://www.jb51.net/article/125076.htm>

a.基本树结构

/\*\*

\* 类名称：Tree

\* 类描述：树形结构（构造多级分类树结构）

\*/

**public** **class** Tree {

**private** String id;

**private** String pId;

**private** String name;

}

b.树形构造工具类

**import** java.util.ArrayList;

**import** java.util.LinkedHashMap;

**import** java.util.List;

**import** java.util.Map;

**import** com.dada.manager.entity.Tree;

// 递归构造树型结构

**public** **class** MenuTreeUtil {

**public** **static** Map<String,Object> *mapArray* = **new** LinkedHashMap<String, Object>();

**public** List<Tree> menuCommon;

**public** List<Object> list = **new** ArrayList<Object>();

**public** List<Object> menuList(List<Tree> menu){

**this**.menuCommon = menu;

**for** (Tree x : menu) {

Map<String,Object> mapArr = **new** LinkedHashMap<String, Object>();

// 最高级节点

**if**(x.getpId().equals("-1")){

mapArr.put("id", x.getId());

mapArr.put("name", x.getName());

mapArr.put("pid", x.getpId());

mapArr.put("childList", menuChild(x.getId()));

list.add(mapArr);

}

}

**return** list;

}

**public** List<?> menuChild(String id){

List<Object> lists = **new** ArrayList<Object>();

**for**(Tree a:menuCommon){

Map<String,Object> childArray = **new** LinkedHashMap<String, Object>();

**if**(a.getpId().equals(id) ){

childArray.put("id", a.getId());

childArray.put("name", a.getName());

childArray.put("pid", a.getpId());

childArray.put("childList", menuChild(a.getId()));

lists.add(childArray);

}

}

**return** lists;

}

}

c.Controller调用：

@RequestMapping("/getTree")

@ResponseBody

**public** ResponseEntity<ResultManage> getTree() {

MenuTreeUtil menuTree = **new** MenuTreeUtil();

List<AuthorityInfo> authorityList = authorityInfoService.listAuthorityInfo(**null**);

Map<String, Object> resultMap = **new** HashMap<>(4);

// 结合业务需求查找树形结构

List<Tree> treeNodes = **new** ArrayList<>(authorityList.size());

**for** (AuthorityInfo authorityInfo : authorityList) {

Tree node = **new** Tree();

node.setId(authorityInfo.getAuthorityId());

node.setpId(authorityInfo.getParentId());

node.setName(authorityInfo.getAuthorityName());

treeNodes.add(node);

}

List<Object> menuList = menuTree.menuList(treeNodes);

// 将数据转换成对应的json字符串形式

resultMap.put("list", menuList);

**return** getJsonResult(resultMap);

}

最终测试结果：

