dingding-web

该项目目前尚未迁移，位于appService源码组中。该项目主要做移动端H5页面，项目中大部分会嵌入到app中，比如一些活动、优惠券等等。其中一大部分已经不再使用，

但没有删除。

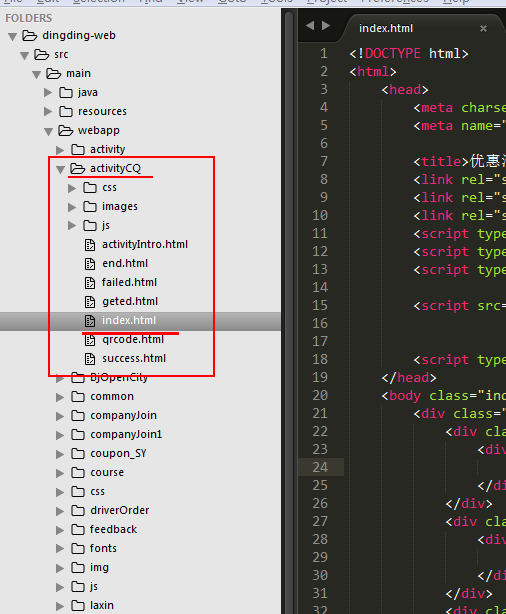
app嵌入页测试

线上地址：

测试地址：<http://test.pdhz-dddc.com/dingding-web/>

可以通过测试链接地址或线上地址+文件夹名字，则默认返回的是这个文件下的index.html 。

如：<http://test.pdhz-dddc.com/dingding-web/> aqctivityCQ/



各个项目模块说明

/dingding-web/src/main/webapp/....

css和js和page目录是一体的，里面涉及到的activity,billing,coupon,user模块没负责过，也不确定线上是否有在使用过。plugin目录存放的是所有需要用到的一些插件，

需要即引用即可。Common目录下的system.js封装了开发中用到的一些公用的方法，这里可能会经常使用到。

1 内嵌在客户端app内的H5页面(正常使用)

course: 使用教程页面 ----内嵌app

feedback: 意见反馈 ---内嵌app

service: 服务页面 ---内嵌app

tourism: 旅游路线 ---内嵌app

userGuides: 用户指南 ---内嵌app

xiecheng: 携程单详情页面 ---内嵌app

webDriverRegister/app: 专车车主认证 ---内嵌app

webXmlPages/404.html: 配置网络异常下的404页面 --内嵌app

rule: 各种协议，隐私说明，法律协议，常见问题等 ---内嵌app

specialCarMoneyRule: 专车计费规则 ---内嵌app

2 内嵌在客户端app内的H5页面(下架过期或未确定的)

laxin: 拉新活动 ---内嵌app(下架)

midAutumnActivity: 中秋活动 ---内嵌app(下架)

renteqsale: 以租代售 ---内嵌app(下架)

shareplan: 共享计划 ---内嵌app(下架)

shengdan: 圣诞元旦双旦活动 ---内嵌app(下架)

shop: 商城 ---内嵌app(下架)

store: 门店 ---内嵌app(下架)

regactive: 免费洗车和搭车现金券(下架)

activity: 老早之前的内嵌app里面的活动页面(不确定线上是否还在使用)

3 提供移动url的H5移动端页面 或 嵌入在微信公众号里面的H5页面

companyJoin: 扫码认证车主 --移动端页面(线上在使用)

superCar: 超级车微信端下单页面 ---移动url(超级车，不确定是否在使用)

companyJoin1: 最老版的扫码认证车主备份(已下架)

activityCQ：重庆优惠券活动页面--内嵌微信公众号(不确定线上是否还在使用)

BjOpenCity: 北京开城优惠券活动页面--内嵌微信公众号(不确定线上是否还在使用)

coupon\_SY: 沈阳优惠券活动--内嵌微信公众号(不确定线上是否还在使用)

driverOrder: 司机订单 ----内嵌在微信公众号里(不确定线上是否还在使用)

recruit: 司机招募移动端页面 ---url访问(不确定线上是否还在使用)

recruit\_pc：司机招募pc端页面 ---url访问(不确定线上是否还在使用)

H5页面开发总结和注意事项

1 通用方法或者格式

每个项目对应自己的目录，每个目录下存放对应的css,js,img目录。

2 ios和android加载web页面的形式

ios和android一般是加载web端提供的项目的路径，然后提供给客户端，客户端在对应的位置加载web的url。或者，客户端(app)请求后台服务器接口，

然后由后台服务器来控制跳转到哪个web页面，这时只需提供web页面的url给服务器即可。

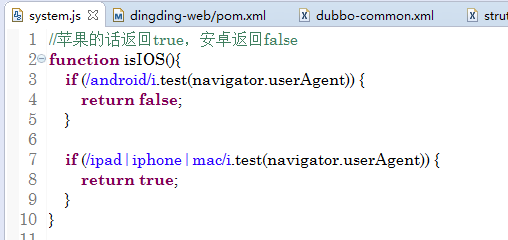
3 客户端和H5数据交互

第一种方式很简单，只适用于web端获取客户端传来的数据。通过在url后面拼接参数，然后才web端获取url后面的参数即可获得从客户端传给web端的部分值。

这种方式只适用于获取单个或少数值，多个即不适用。

第二种方式比较通用，既可以获取客户端传过来的值，又可以将值传给客户端，项目中一般采用这种方法。需要分ios和android来操作js。Common目录下封装了js

区分ios和android的方法，直接调用即可。/dingding-web/src/main/webapp/common/js/system.js

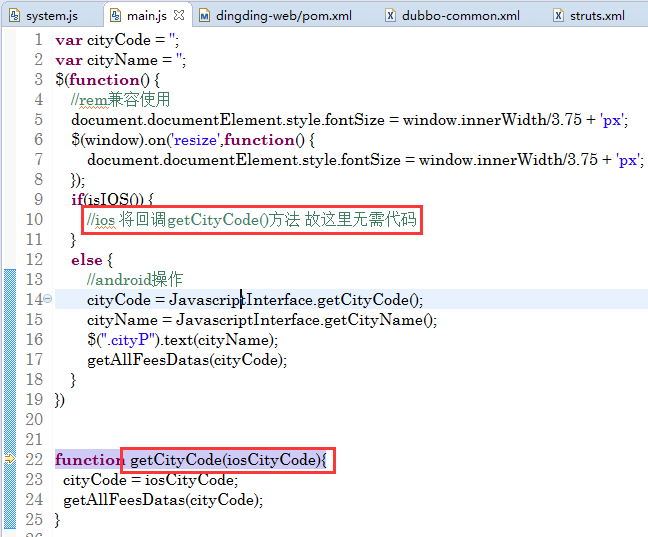


4 web端获取客户端传过来的值

**IOS**下，首先双方约定一个方法，这个方法的功能是需要获取ios端传过来的某些值，然后在js中全局定义这样一个方法，然后客户端会在加载url的时候调用js里面的

这个方法(真正意义上是ios在加载web url的时候回进行一个回调函数的调用，这个回调函数就是在js里面定义好的。)，然后我们就可以在js里定义的这个方法里面通过参数获取到客户端传过来的参数值。

eg: /dingding-web/src/main/webapp/specialCarMoneyRule/main.js



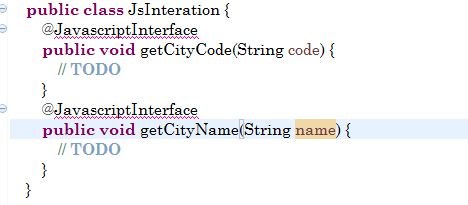
function getCityCode()就是IOS客户端加载web url的时候会回调的方法，只需要在js中定义这个回调函数，然后通过参数来获取IOS客户端传过来的值。

Android下,相对来说比较简单。只需要在客户端定义一个方法，然后这个方法返回web端需要的值，然后在web端就可以通过JavascriptInterface这个对象去调用客户端定义的方法，就可以取到值。

在上图中，cityCode = JavascriptInterface.**getCityCode()**; cityName = JavascriptInterface.**getCityName()**; 这两个方法是在Android客户端中定义的两个方法，每个方法对应返回web需要的值。

JavascriptInterface是Android下的一个WebView设置供JavaScript调用的交互接口。android.webkit.JavascriptInterface;

eg：java代码如下

 更详细的用法请参考：<http://www.imooc.com/article/1475>

5 web端传值给客户端

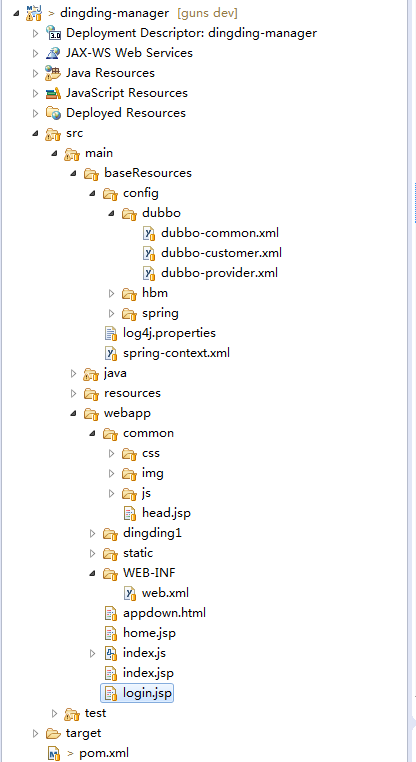
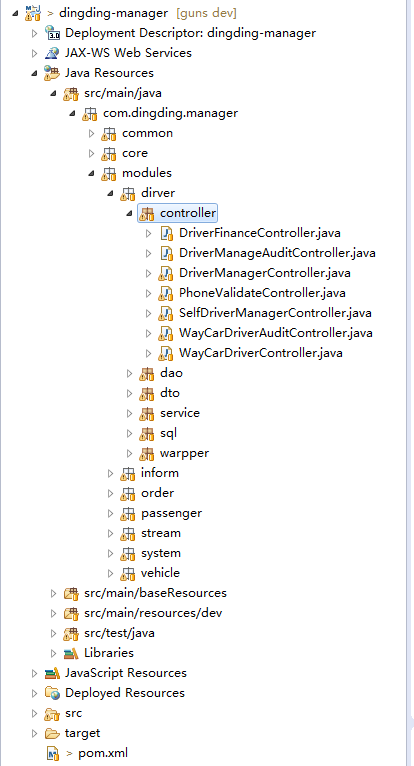
ios下,调用window.webkit.messageHandlers.iosMethod.postMessage(Object)即可，这里的iosMethod是ios客户端定义好的方法，然后通过调用这个方法的postMessage方法就能将值传给客户端，参数Object是一个json对象，客户端需要接收哪些字段值。

android下，通过调用JavascriptInterface.androidMethod(a,b,c)即可，这里的androidMethod是android客户端下定义好的方法用来接收web端传过去的值，需要哪些就传几个参数。

eg: //dingding-web/src/main/webapp/js/activity/sharecode.js 页面分享功能



dingding-manager 运管系统交互说明

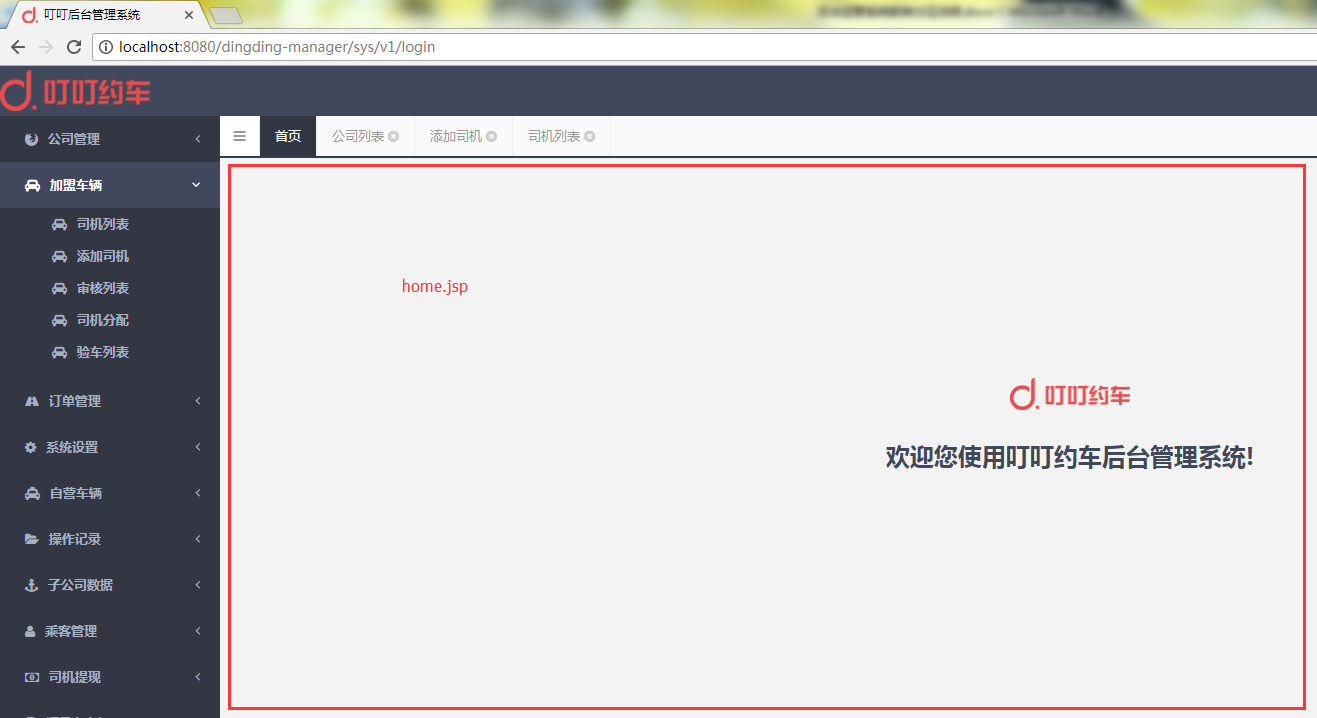


页面结构

1.Login.jsp--->登录页面

2.Index.jsp--->主要包括登录成功之后的左侧菜单和整个页面的头部和尾部。内嵌了iframe标签，默认为home.jsp。

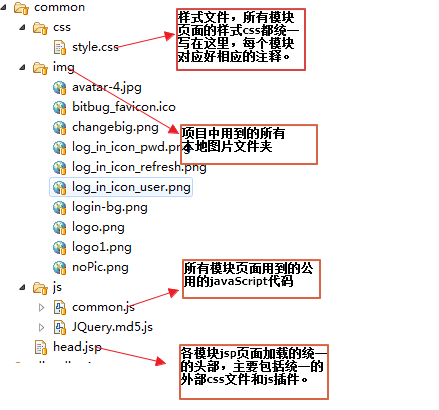
3.home.jsp--->默认登录进来呈现给用户的内容页面



前端资源目录

common目录

主要存放各模块公用的css,js,img和头部head.jsp文件。



static目录

存放项目中可能会用到的相关外部插件。

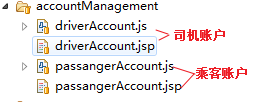
dingding1/modules

项目的各个模块



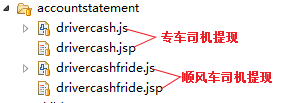
1 accountManagement

账户管理模块



2 accountstatement

流水对账模块



3 childcompany

子公司数据模块



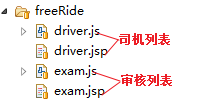
4 company

公司管理模块



5 freeRide

顺风车模块



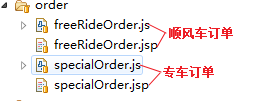
6 logs

操作记录模块



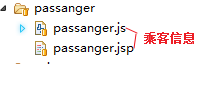
7 order

订单管理模块



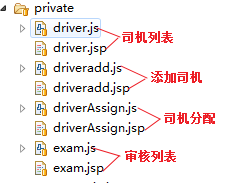
8 passanger

乘客管理模块



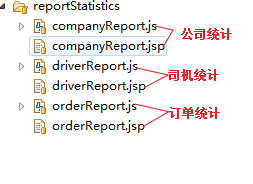
9 private

加盟车辆模块



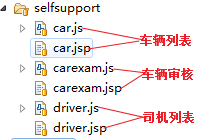
10 reportStatistics

报表统计模块



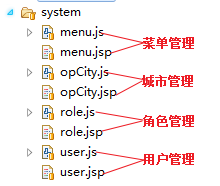
11 selfsupport

自营车辆模块



12 system

系统设置模块



新增模块和在已有模块中新增列表项

新增模块只需在dingding1/modules下面新建模块名和对应的列表项，然后通过菜单管理配置在项目中。最后通过角色管理进行授权展示出来即可。左侧的菜单列表以及每

个列表里面的按钮都是存放在数据库中的。左侧的菜单列表在index.js中通过请求后台接口返回过来的。

jquery-dataTable分页表格使用说明

使用jquery-dataTable插件需要在相应的jsp页面引入相应的css和js文件。都写在公共的头部head.jsp中。

<link rel=*"stylesheet"* href=*"*${ctx}*/static/dataTables/css/jquery.dataTables.min.css"* />

<script type=*"text/javascript"*src=*"*${ctx}*/static/dataTables/jquery.dataTables.js"*></script>

直接按下面这种方法写好表头，定义好table的id。



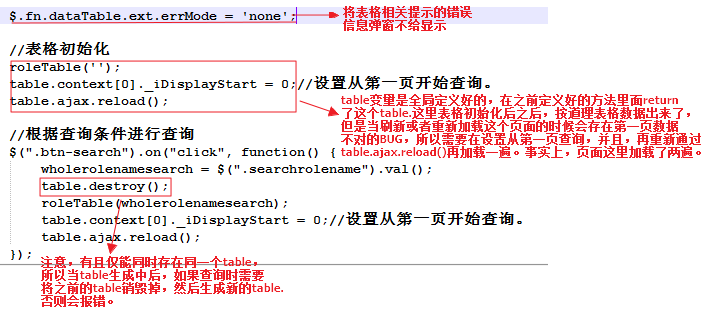
在每个列表页定义好自己的初始化表格函数，然后在里面处理请求条件和对表格每列的处理。

 “dataSrc”:”aaData” 是约定的。

对每列的处理：



在js里面调用之前定义好的表格方法来进行初始化，以及根据查询条件来查询。



bootstrap模态框

1 modal使用说明

页面中所有的增删改查弹出来的模态框都是通过bootstrap的模态框来实现的。通用的模态框的代码如下：



2 常用的模态框的操作

$("#myModalEdit").modal('show');//模态框显示

$("#myModalEdit").modal('hide');//模态框隐藏

$('#myModalEdit').on('hide.bs.modal', **function** () {

// 模态框隐藏消失触发的事件操作....

});

表格操作列的相关详情，编辑，删除操作

1 点击操作列下的某个按钮都能取到每行的具体数据

$(document).on("click", ".btn-detail", **function**() {

// 可以取得后台返回列表页每行的所有数据

**var** data = table.row( $(**this**).parents('tr') ).data();

});

2 详情和编辑都是走接口取到相关的详情数据渲染在对应的模态框上

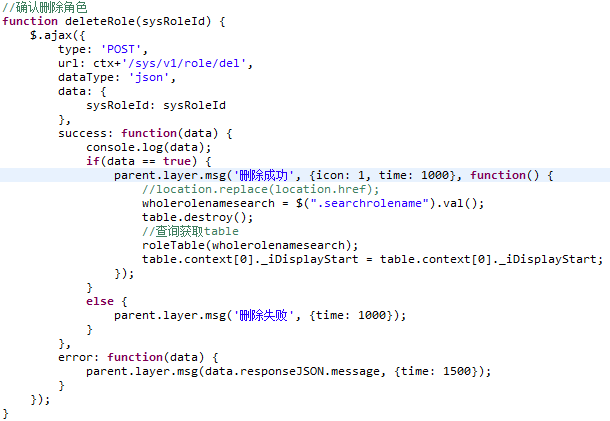
3 删除和启用禁用操作都调用了layer.js插件来确认。



每个操作凡是涉及到请求后台接口的操作（比如确认修改，确认删除，确认授权，启用禁用，审核等操作）都需要重新根据查询条件以及当前列表所处的页数来重新渲染列表。

以此来实现实时保存当前分页状态并不影响上次查询结果。

Eg：



shiro标签控制每个页面的按钮

1 jsp页面的头部引入

<%@taglib prefix=*"shiro"* uri=*"http://shiro.apache.org/tags"* %>

2 在相应的jsp页面需要按钮的地方加上控制

<shiro:hasPermission name=*'sys:driverReport:view'*>

<button class=*"btn search-btn btn-info"* style="padding: *0 10px*;">

<i class=*'fa fa-search'*></i>查询

</button>

</shiro:hasPermission>

3 最后这个按钮添加到数据库中有一个权限标识，通过name属性和这个权限标识绑定即可。

表单字段验证bootstrapValidator

1 HTML元素验证

正常情况下，需要在对应的jsp页面引入相应的css和js文件，我们都统一写在head.jsp中。Jsp页面中在需要验证的字段模块中用

<form id="addroleform" class="form-horizontal"></form>

包起来,然后通过input的name属性绑定。通过id来绑定。

Eg：



2 Js中进行验证

