**物联网应用层HTML5+ APP端与后端交互**

**1.交互方式：$.ajax**

1）$.ajax封装了XmlHttpRequest这个对象，实际发送的还是遵循HTTP协议的HTTP请求

2）前后端约定参数数据是JSON对象和结果数据的格式是JSON字符串，又因为请求报文必须是字符串形式，因此发送参数时需要先用JSON.stringify转化，$.ajax会自动将data声明的参数数据放在消息体中（POST请求时）或者url中（GET请求时）

3）$.ajax只是个ajax工具，把参数的设置分离，且提供了回调函数来处理响应返回的数据

**2.接口示例：**

**H5 APP端的ajax：（H5 APP端可以看成前端）**

$.ajax({

type:"post",

// queryType暂时为1

url:getRootPath() + "rest/factory/XpFactoryTaskService/findUserTaskList?token="+token+"&page=1&queryType=1",

success:function(data){

………………

**后端的serviceImpl.java：**

@MethodParameter(desc="findUserTaskList", input="user,keyword,queryType,page",userParam = "user")

@Override

**public** List<XpFactoryTask> findUserTaskList(User user,String keyword,String queryType,**int** page) **throws** Exception {

// 拿到每页显示条数

**int** pageCount = 0;

String value = xpParamService.queryValueByCode("PAGE\_SIZE", user);

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

pageCount = Integer.*parseInt*(value);

}

Map params = **new** HashMap<String,String>();

params.put("keyword", keyword);

List<XpFactoryTask> xpFactoryTasks = **null**;

**if**(!StringUtils.*isEmpty*(queryType)){

**if**("1".equals(queryType)){

//查报案推，定损推

xpFactoryTasks = xpFactoryTaskDao.findNoticListByUser(params,user,pageCount,page);

}

**else**{

String taskStatus = **null**;

**if**(queryType.equals("2")){

taskStatus = TaskStatus.***WAIT\_FOR\_GET***;

}

**else** **if**(queryType.equals("3")){

taskStatus = TaskStatus.***REPAIRING***;

}

**else** **if**(queryType.equals("4")){

taskStatus = TaskStatus.***FINISH\_REPAIR***;

}

**else** **if**(queryType.equals("5")){

taskStatus = TaskStatus.***GIVE\_CAR***;

}

**else** **if**(queryType.equals("6")){

taskStatus = TaskStatus.***BACK***;

}

//定待接

xpFactoryTasks = xpFactoryTaskDao.findWaitForGetList(params, user,taskStatus,pageCount,page);

}

}**else**

{

xpFactoryTasks = xpFactoryTaskDao.findAll(params, user,pageCount,page);

}

//格式化数据

xpFactoryTasks = DataFormatUtil.*formatFactoryTaskList*(xpFactoryTasks);

**return** xpFactoryTasks;

}

**serviceImpl中的exeMethod方法：**

@SuppressWarnings("unchecked")

@Override

**public** Object exeMethod(IBaseService baseService,Method method, Object[] args) **throws** Exception {

//提交事务注解：@Transactional(rollbackFor=Exception.class)

XpFactoryTaskService service=(XpFactoryTaskService)baseService;

**if** (method.getName().equals("findUserTaskList")){

//@MethodParameter(desc="findUserTaskList", input="user,keyword,queryType,page",postType={},postName="",queryString="")

**return** service.findUserTaskList((User)args[0],(String)args[1],(String)args[2],(Integer)args[3]);

}

**if** (method.getName().equals("backTask")){

//@MethodParameter(desc="backTask", input="backReasonCode,backRemark,taskId,user",postType={},postName="",queryString="")

**return** service.backTask((String)args[0],(String)args[1],(String)args[2],(User)args[3]);

}

**if** (method.getName().equals("getCar")){

//@MethodParameter(desc="getCar", input="taskId,user",postType={},postName="",queryString="")

**return** service.getCar((String)args[0],(User)args[1]);

}

**if** (method.getName().equals("updateExpectTime")){

//@MethodParameter(desc="updateExpectTime", input="taskId,expectTime,isMsgNotic,msgContext,user",postType={},postName="",queryString="")

**return** service.updateExpectTime((String)args[0],(String)args[1],(String)args[2],(String)args[3],(User)args[4]);

}

**if** (method.getName().equals("finishRepair")){

//@MethodParameter(desc="finishRepair", input="taskId,user",postType={},postName="",queryString="")

**return** service.finishRepair((String)args[0],(User)args[1]);

}

**if** (method.getName().equals("updateBespeakTime")){

//@MethodParameter(desc="updateBespeakTime", input="taskId,bespeakTime,isMsgNotic,msgContext,user",postType={},postName="",queryString="")

**return** service.updateBespeakTime((String)args[0],(String)args[1],(String)args[2],(String)args[3],(User)args[4]);

}

**if** (method.getName().equals("giveCar")){

//@MethodParameter(desc="giveCar", input="taskId,user",postType={},postName="",queryString="")

**return** service.giveCar((String)args[0],(User)args[1]);

}

**return** **super**.exeMethod(baseService,method, args);

}

**后端的serviceapi.xml：（service的配置文件）**

<service serviceName=*"XpFactoryTaskService"* className=*"com.dstech.dssp.factory.service.XpFactoryTaskService"* serviceType=*"service"*>

<serviceDesc>XpFactoryTaskService</serviceDesc>

<webPage methodName=*"xpfactorytaskListPage"* pageDesc=*"列表"* viewName=*"busicomps/factory/content/xpfactorytask/xpfactorytaskList"*/>

<webPage methodName=*"xpfactorytaskFormPage"* pageDesc=*"表单"* viewName=*"busicomps/factory/content/xpfactorytask/xpfactorytaskForm"*/>

<webPage methodName=*"xptaskInfoPage"* pageDesc=*"展示案件详情"* viewName=*"busicomps/factory/content/xpfactorytask/taskInfo"*/>

<serviceMethod name=*"findUserTaskList"* desc=*"findUserTaskList"* url=*"/factory/XpFactoryTaskService/findUserTaskList"*>

<serviceParam dataType=*"User"* paramName=*"user"* user=*"true"*/>

<serviceParam dataType=*"String"* paramName=*"keyword"*/>

<serviceParam dataType=*"String"* paramName=*"queryType"*/>

<serviceParam dataType=*"int"* paramName=*"page"*/>

<returnParam dataType=*"List&lt;XpFactoryTask&gt;"*/>

<exceptionParams>

<exceptionParam dataType=*"Exception"*/>

</exceptionParams>

</serviceMethod>

<serviceMethod name=*"backTask"* desc=*"backTask"* url=*"/factory/XpFactoryTaskService/backTask"*>

<serviceParam dataType=*"String"* paramName=*"backReasonCode"*/>

<serviceParam dataType=*"String"* paramName=*"backRemark"*/>

<serviceParam dataType=*"String"* paramName=*"taskId"*/>

<serviceParam dataType=*"User"* paramName=*"user"* user=*"true"*/>

<returnParam dataType=*"java.util.Map"*/>

<exceptionParams>

<exceptionParam dataType=*"Exception"*/>

</exceptionParams>

</serviceMethod>

<serviceMethod name=*"getCar"* desc=*"getCar"* url=*"/factory/XpFactoryTaskService/getCar"*>

<serviceParam dataType=*"String"* paramName=*"taskId"*/>

<serviceParam dataType=*"User"* paramName=*"user"* user=*"true"*/>

<returnParam dataType=*"java.util.Map"*/>

<exceptionParams>

<exceptionParam dataType=*"Exception"*/>

</exceptionParams>

</serviceMethod>

<serviceMethod name=*"updateExpectTime"* desc=*"updateExpectTime"* url=*"/factory/XpFactoryTaskService/updateExpectTime"*>

<serviceParam dataType=*"String"* paramName=*"taskId"*/>

<serviceParam dataType=*"String"* paramName=*"expectTime"*/>

<serviceParam dataType=*"String"* paramName=*"isMsgNotic"*/>

<serviceParam dataType=*"String"* paramName=*"msgContext"*/>

<serviceParam dataType=*"User"* paramName=*"user"* user=*"true"*/>

<returnParam dataType=*"java.util.Map"*/>

<exceptionParams>

<exceptionParam dataType=*"Exception"*/>

</exceptionParams>

</serviceMethod>

<serviceMethod name=*"finishRepair"* desc=*"finishRepair"* url=*"/factory/XpFactoryTaskService/finishRepair"*>

<serviceParam dataType=*"String"* paramName=*"taskId"*/>

<serviceParam dataType=*"User"* paramName=*"user"* user=*"true"*/>

<returnParam dataType=*"java.util.Map"*/>

<exceptionParams>

<exceptionParam dataType=*"Exception"*/>

</exceptionParams>

</serviceMethod>

<serviceMethod name=*"updateBespeakTime"* desc=*"updateBespeakTime"* url=*"/factory/XpFactoryTaskService/updateBespeakTime"*>

<serviceParam dataType=*"String"* paramName=*"taskId"*/>

<serviceParam dataType=*"String"* paramName=*"bespeakTime"*/>

<serviceParam dataType=*"String"* paramName=*"isMsgNotic"*/>

<serviceParam dataType=*"String"* paramName=*"msgContext"*/>

<serviceParam dataType=*"User"* paramName=*"user"* user=*"true"*/>

<returnParam dataType=*"java.util.Map"*/>

<exceptionParams>

<exceptionParam dataType=*"Exception"*/>

</exceptionParams>

</serviceMethod>

<serviceMethod name=*"giveCar"* desc=*"giveCar"* url=*"/factory/XpFactoryTaskService/giveCar"*>

<serviceParam dataType=*"String"* paramName=*"taskId"*/>

<serviceParam dataType=*"User"* paramName=*"user"* user=*"true"*/>

<returnParam dataType=*"java.util.Map"*/>

<exceptionParams>

<exceptionParam dataType=*"Exception"*/>

</exceptionParams>

</serviceMethod>

</service>

**后端的XpFactoryTask.xml（bean的配置文件）**

<?xml version=*"1.0"* encoding=*"utf-8"* standalone=*"yes"*?>

<bean id=*"XpFactoryTask"* beanClass=*"com.dstech.dssp.factory.bean.XpFactoryTask"* nameSpace=*"factory"* daoClass=*"com.dstech.dssp.factory.dao.XpFactoryTaskDao"* serviceClass=*"com.dstech.dssp.factory.service.XpFactoryTaskService"* ormType=*"bomf"* xmlns=*"http://www.dstech.com/schema/bean"*>

<viewConfig>

<beanView>busicomps/factory/content/xpfactorytask/xpfactorytaskForm</beanView>

<listView>factory/XpFactoryTaskService/xpfactorytaskListPage</listView>

</viewConfig>

<fieldMappings>

<fieldMapping beanField=*"id"* sqlField=*"id"* title=*"id"* jdbcType=*"2"* dataType=*"string"*/>

<fieldMapping beanField=*"lpTaskId"* sqlField=*"lp\_task\_id"* title=*"lpTaskId"* jdbcType=*"12"* dataType=*"string"*/>

<fieldMapping beanField=*"registNo"* sqlField=*"regist\_no"* title=*"registno"* jdbcType=*"12"* dataType=*"string"*/>

<fieldMapping beanField=*"linkMan"* sqlField=*"link\_man"* title=*"linkMan"* jdbcType=*"12"* dataType=*"string"*/>

<fieldMapping beanField=*"linkTel"* sqlField=*"link\_tel"* title=*"linkTel"* jdbcType=*"12"* dataType=*"string"*/>

<fieldMapping beanField=*"factoryCode"* sqlField=*"factory\_code"* title=*"factoryCode"* jdbcType=*"12"* dataType=*"string"*/>

<fieldMapping beanField=*"factoryName"* sqlField=*"factory\_name"* title=*"factoryName"* jdbcType=*"12"* dataType=*"string"*/>

<fieldMapping beanField=*"taskStatus"* sqlField=*"task\_status"* title=*"taskStatus"* jdbcType=*"12"* dataType=*"string"*/>

<fieldMapping beanField=*"licenseNo"* sqlField=*"license\_no"* title=*"licenseNo"* jdbcType=*"12"* dataType=*"string"*/>

<fieldMapping beanField=*"acceptTime"* sqlField=*"accept\_time"* title=*"acceptTime"* jdbcType=*"93"* dataType=*"date"*/>

<fieldMapping beanField=*"vinNo"* sqlField=*"vin\_no"* title=*"vinNo"* jdbcType=*"12"* dataType=*"string"*/>

<fieldMapping beanField=*"wfid"* sqlField=*"wfid"* title=*"wfid"* jdbcType=*"12"* dataType=*"string"*/>

<fieldMapping beanField=*"wfStatus"* sqlField=*"wf\_status"* title=*"wfStatus"* jdbcType=*"12"* dataType=*"string"*/>

<fieldMapping beanField=*"wfStatusTime"* sqlField=*"wf\_status\_time"* title=*"wfStatusTime"* jdbcType=*"93"* dataType=*"date"*/>

<fieldMapping beanField=*"wfBack"* sqlField=*"wf\_back"* title=*"wfBack"* jdbcType=*"2"* dataType=*"long"*/>

<fieldMapping beanField=*"wfStatusExt"* sqlField=*"wf\_status\_ext"* title=*"wfStatusExt"* jdbcType=*"12"* dataType=*"string"*/>

<fieldMapping beanField=*"createTime"* sqlField=*"create\_time"* title=*"createTime"* jdbcType=*"93"* dataType=*"date"*/>

<fieldMapping beanField=*"modifyTime"* sqlField=*"modify\_time"* title=*"modifyTime"* jdbcType=*"93"* dataType=*"date"*/>

<fieldMapping beanField=*"orgId"* sqlField=*"org\_id"* title=*"orgId"* jdbcType=*"12"* dataType=*"string"*/>

<fieldMapping beanField=*"scheduleId"* sqlField=*"schedule\_id"* title=*"scheduleId"* jdbcType=*"12"* dataType=*"string"*/>

<fieldMapping beanField=*"comCode"* sqlField=*"com\_code"* title=*"comCode"* jdbcType=*"12"* dataType=*"string"*/>

<fieldMapping beanField=*"lflag"* sqlField=*"lflag"* title=*"lflag"* jdbcType=*"12"* dataType=*"string"*/>

<fieldMapping beanField=*"taskType"* sqlField=*"task\_type"* title=*"taskType"* jdbcType=*"12"* dataType=*"string"*/>

</fieldMappings>

<beanSql>select \* from XP\_FACTORY\_TASK </beanSql>

<pkField fieldNames=*"ID"* dataTypes=*"string"* beanAttributes=*"id"*/>

<saveConfig tableName=*"XP\_FACTORY\_TASK"*>

<preSaveDefaults>

<preSaveDefault attributeName=*"id"* updateType=*"insert"* gloabVariable=*"uuid"*/>

<preSaveDefault attributeName=*"createTime"* updateType=*"insert"* gloabVariable=*"sysdate"*/>

<preSaveDefault attributeName=*"modifyTime"* updateType=*"insert"* gloabVariable=*"sysdate"*/>

<preSaveDefault attributeName=*"orgId"* updateType=*"insert"* gloabVariable=*"orgid"*/>

<preSaveDefault attributeName=*"modifyTime"* updateType=*"update"* gloabVariable=*"sysdate"*/>

</preSaveDefaults>

</saveConfig>

<querys>

<query id=*"queryList"*>

<sql>org\_id=?</sql>

<sqlParams>

<sqlParam dataType=*"string"* variableName=*"orgid"*/>

</sqlParams>

</query>

</querys>

<queryFilters>

<filter filterName=*"licenseNo"* filterSql=*"license\_no like ?"* dataType=*"string"* likeType=*"LIKE"*/>

<filter filterName=*"registNo"* filterSql=*"regist\_no like ?"* dataType=*"string"* likeType=*"LIKE"*/>

<filter filterName=*"id"* filterSql=*"ID=?"* dataType=*"string"*/>

<filter filterName=*"id1"* filterSql=*"ID=?"* dataType=*"string"*/>

</queryFilters>

</bean>

3.再谈前后端交互链

**参与交互的有：**

前端的ajax

后端的serviceImpl.java

后端的serviceapi.xml

后端的bean.xml（更多的还是被DAO解析）

**交互链：**

前端的$.ajax发出ajax请求---🡪

所有url被DispatcherServlet拦截---🡪

url通过RequestMapping映射到DispatcherServlet中具体的service（反射）----🡪

具体的service通过参数找到具体的方法（通过exeMethod方法匹配）----🡪

具体的方法处理参数返回数据（通常是List或者Map）-----🡪

DispatcherServlet中的service对返回数据进行转化，转成JSON字符串写入响应---🡪

前端的$.ajax接受ajax响应---🡪

$.ajax解析响应取出响应结果封装到新的data数组中----🡪

data作为形参传入回调函数，由其中的业务逻辑进行处理

**细节疑问：**

1.serviceapi.xml怎么起作用

2.bean.xml怎么起作用

3.谁来将参数通过注解MethodParameter注入方法

**对于这三个问题我们先来看下serviceImpl.java中的exeMethod：**

**public** Object exeMethod(IBaseService baseService,Method method, Object[] args) **throws** Exception {

//提交事务注解：@Transactional(rollbackFor=Exception.class)

XpFactoryTaskService service=(XpFactoryTaskService)baseService;

**if** (method.getName().equals("findUserTaskList")){

//@MethodParameter(desc="findUserTaskList", input="user,keyword,queryType,page",postType={},postName="",queryString="")

**return** service.findUserTaskList((User)args[0],(String)args[1],(String)args[2],(Integer)args[3]);

}

………………

可以看到exeMethod接受的参数已经是被合理安排好了的，这部分应该是在DispatcherServlet中对请求的参数进行解析，然后分别实例化成IBaseService对象。Method对象、Object[]对象。（这部分可能使用**反射**的技术 ）

而DispatcherServlet是怎么从报文的字符串就得到这些对象的呢？

答案是根据serviceapi.xml和bean.xml中的部分信息知道应该根据什么信息创建什么对象。当然是反射创建的。

DispatcherServlet创建完对象后，就把这些参数对应地传给了exeMethod方法，然后就是exeMethod来根据逻辑调用方法了。

**下面我们来模拟一下DispatcherServlet的处理：**

doGet(…){

…

//根据serviceapi.xml解析请求的url将请求分派到对应service处理的代码

…

//通用的service处理方法

//根据serviceapi.xml、bean.xml解析参数创建对象

//调用service中的exeMethod方法，传入响应对象作为参数

…

}

doPost(…){

…

doGet(…);

…

}

当然，具体的细节还要看平台的DispatcherServlet是怎么操作的，应该不会有太大的偏差。

记住一点，DD就是用来解析的，目的是为了方便修改配置，不然在这里就必须写在DispatcherServlet中的，那么就必须为每个service创建一个解析，由于大量代码冗余，我们直接的思路就是一个service处理方法里面进行多个条件判断，但是这样又缺失了灵活性，需求更改时需要改动大量代码，于是我们用一个service处理方法加上DD来处理这种冗余。

**还有一个疑问：MethodParameter是exeMethod里那样直接传的吗？**

这涉及到注解的正确打开方式！