开发环境配置及说明

# 环境安装及环境变量配置

## 开发环境拷贝

从svn上『17-开发环境』下载devenv和devsoft文件夹并拷贝到D盘，并对压缩包进行解压缩。以下的环境配置皆是以D:\devenv为根路径进行配置。



## 环境变量配置

### 新建环境变量JAVA\_HOME

JAVA\_HOME=D:\devenv\Java\jdk1.8.0\_101

### 新建环境变量CLASSPATH

CLASSPATH=%JAVA\_HOME%\lib\dt.jar;%JAVA\_HOME%\lib\tools.jar;

### 在PATH变量中添加路径

%JAVA\_HOME%\bin;%JAVA\_HOME%\jre\bin;D:\devenv\nodejs\;D:\devenv\ruby-2.2.3-x64-mingw32\bin;D:\devenv\Python27;D:\devenv\nodejs\npm\_global;D:\devenv\PortableGit-2.8.3-64-bit.7z\cmd

## 测试环境变量是否配置成功

在cmd窗口下

npm –v



查看版本号如成功，则node npm安装成功

npm install npm -g

npm install grunt -g

npm install bower -g

npm install node-gyp -g

npm install http-server -g

python如有问题，请用安装包重新安装到D:\nodejs\Python27

python –v

# 前端配置

## 修改配置文件

因为在开发过程中会遇到内网/外网的问题，基于此对npmrc文件做了对内网/外网环境的不同配置

打开D:\devenv\nodejs\etc\npmpc文件，根据网络环境对其内容进行修改。

### 外网环境

外网可以是用淘宝的npm源，npmrc文件如下

registry=https://registry.npm.taobao.org/

prefix=D:\devenv\nodejs\npm\_global

cache=D:\devenv\nodejs\npm\_cache

### 内网环境

如果使用内网环境需要在本地服务器上建立本地的私服。linux可以通过nexus建立私服。

那么对应的源地址配置修改为本地私服即可

registry=http://172.16.17.220:8081/repository/npm-group/

prefix=D:\devenv\nodejs\npm\_global

cache=D:\devenv\nodejs\npm\_cache

## 前端应用部署

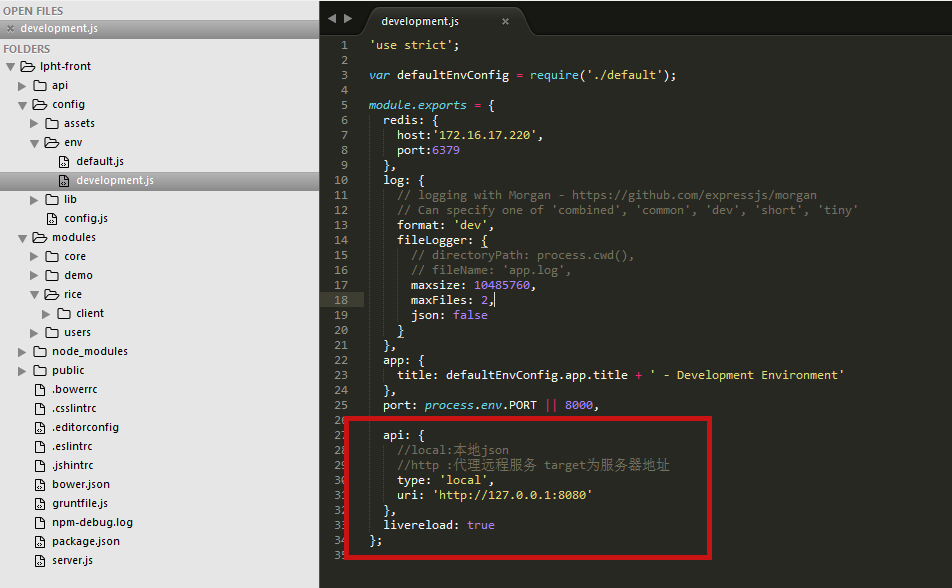
通过命令行进入工程文件目录，..\trunk\frontend\lpht-front，在命令行执行以下命令

npm install

bower install

## 修改前端模式开发模式

打开..\trunk\frontend\lpht-front\config\env\development.js



Type：local读取本地json，http为代理远程服务；

Uri：为服务器地址。

## 启动服务

在命令行模式下，进入..\trunk\frontend\lpht-front目录下，运行grunt



启动完成后，打开浏览器，输入<http://127.0.0.1:8000/>，若能访问成功，则说明前端环境部署成功。

# 后端配置

## 修改本地maven库配置

打开..\trunk\backend\jldp-parent\pom.xml

<repository>

<id>maven-public</id>

<name>maven-public</name>

<url>http://172.16.17.220:8081/repository/maven-public</url>

<snapshots>

<enabled>true</enabled>

</snapshots>

<releases>

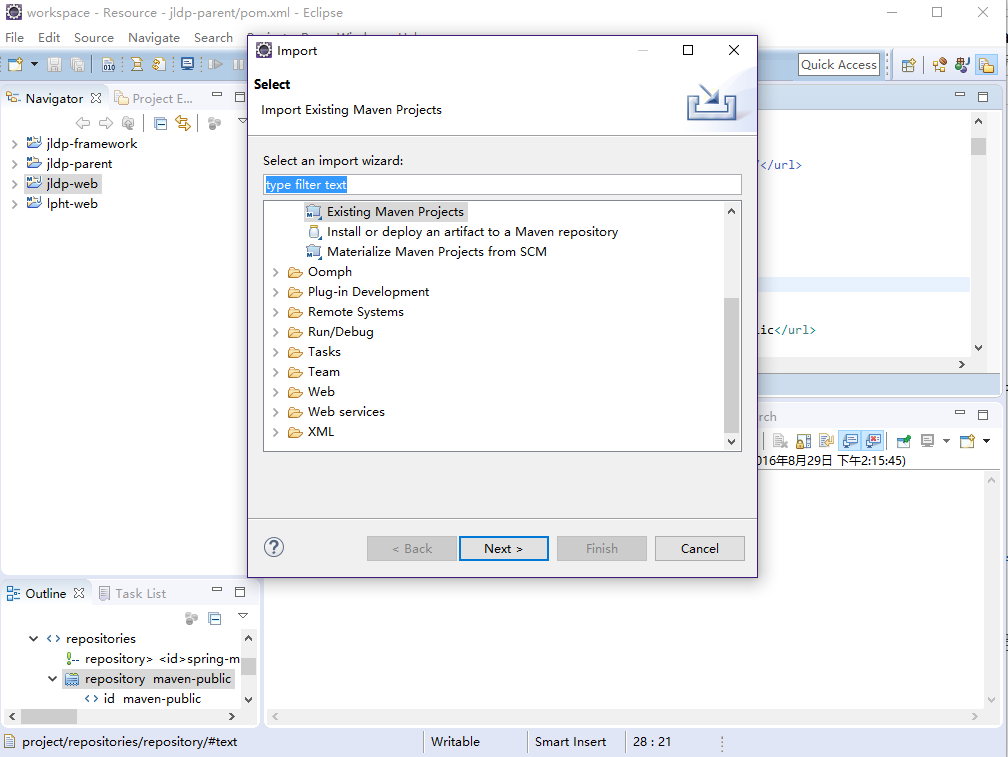
<enabled>true</enabled>

</releases>

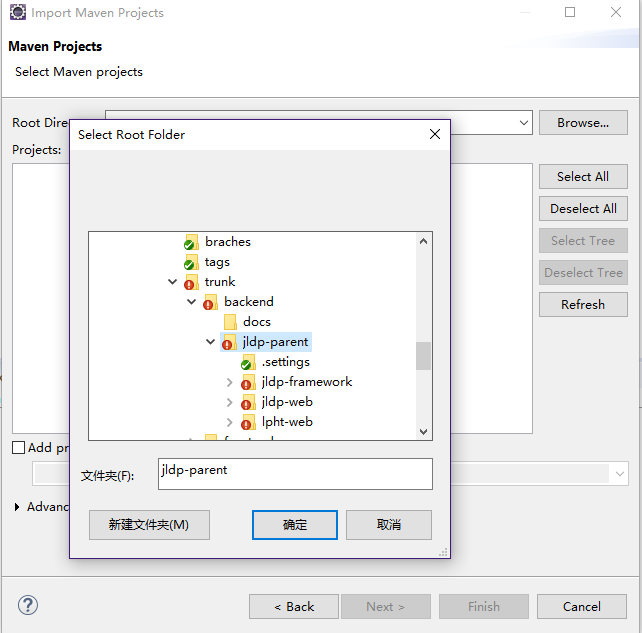
</repository>

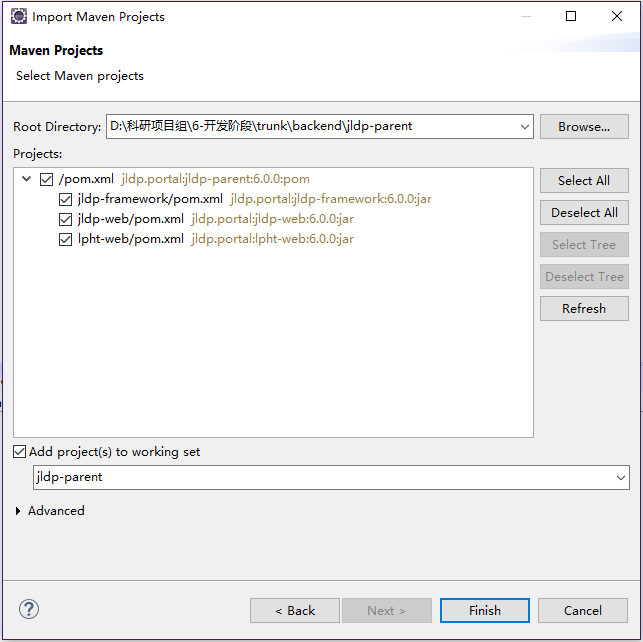
本项目的本地库路径指向<http://172.16.17.220:8081/repository/maven-public>，根据实际情况进行修改，若无变动则无需修改。

## 导入项目工程文件



初次导入工程时需从maven库往本地加载jar包，请耐心等待。





## 工程说明

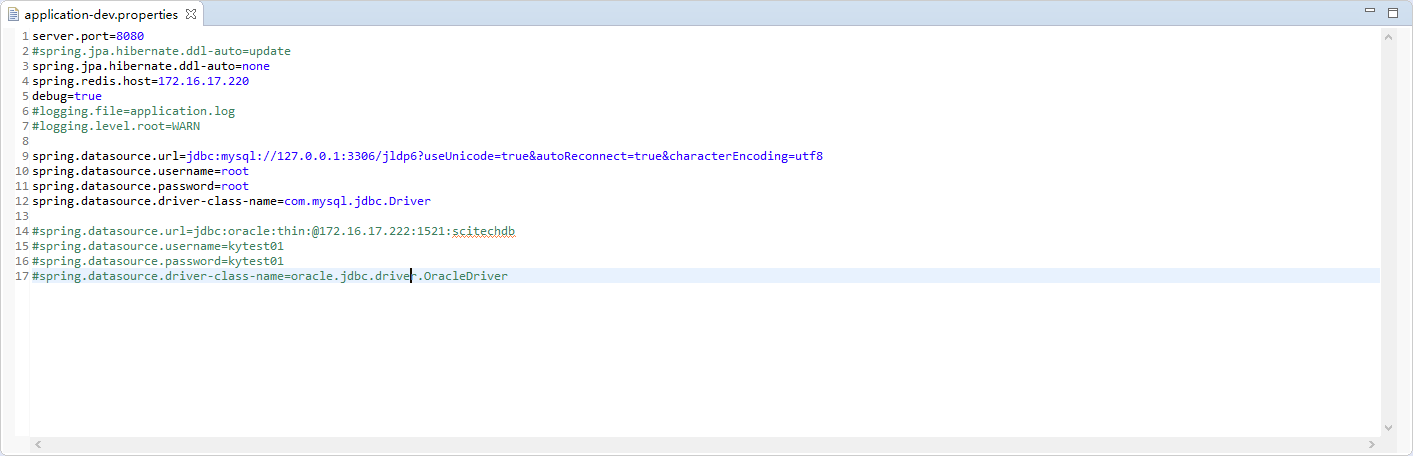
jldp-framework ：系统框架

jldp-web：组织结构、角色、权限等系统级功能

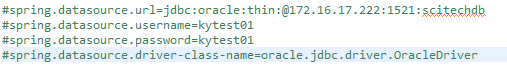
lpht-web：业务功能

## 修改数据库连接配置

打开/jldp-web/src/main/resources/application-dev.properties文件



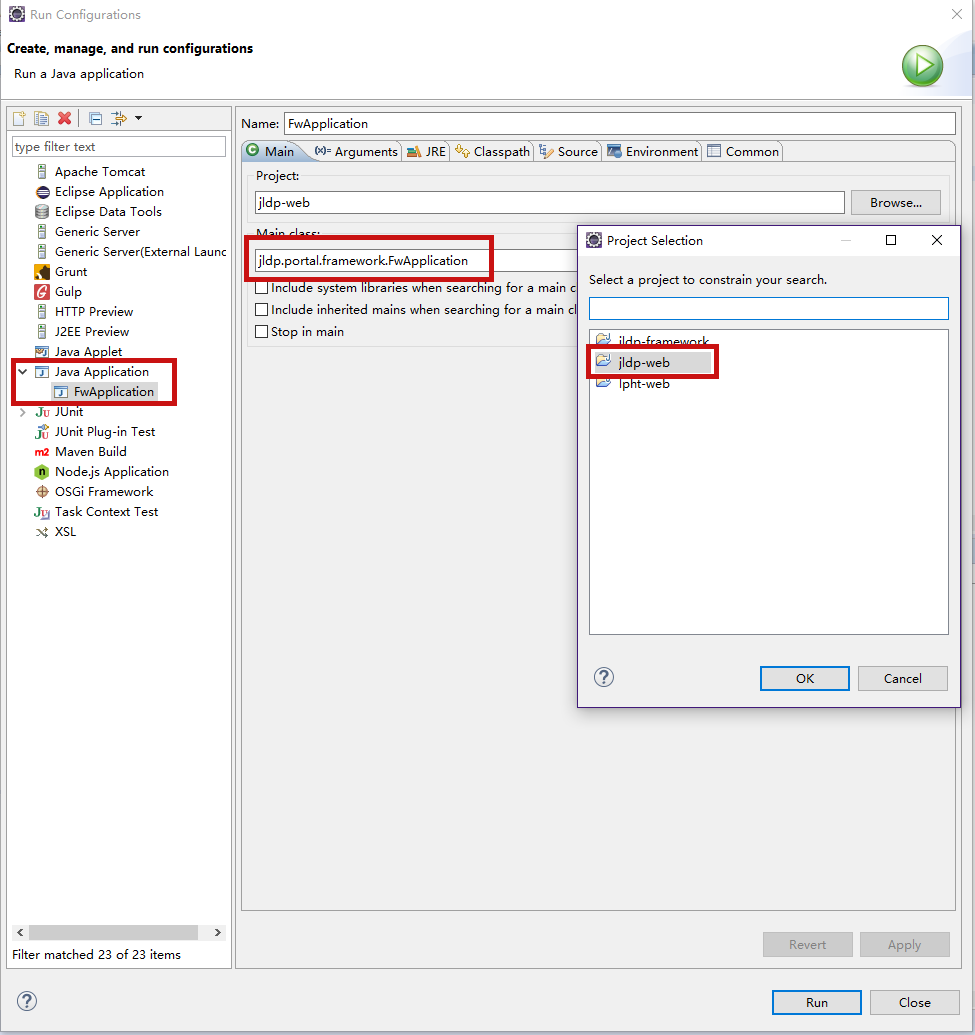
Oracle数据库配置



Mysql数据库配置

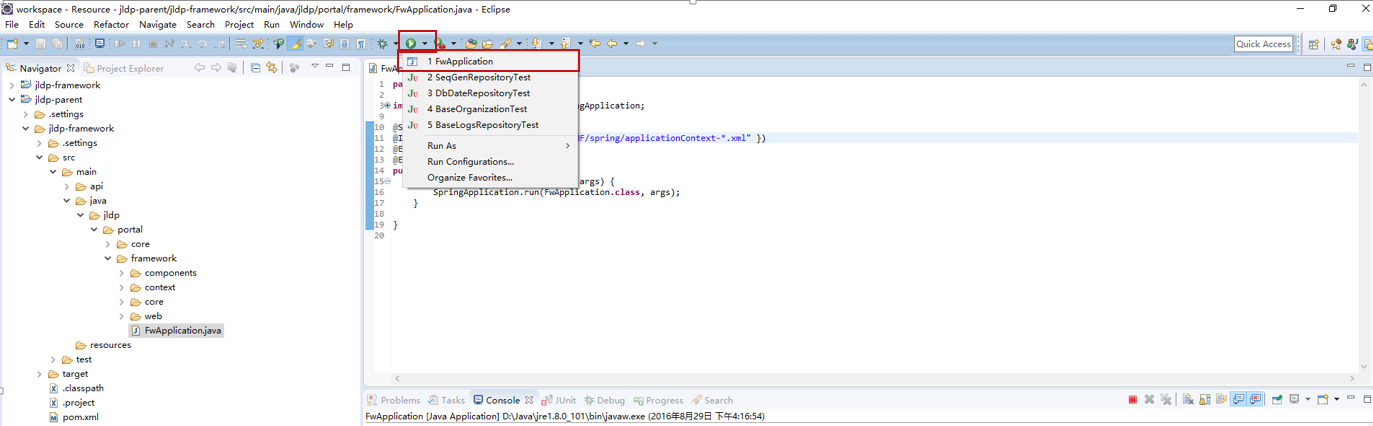


## 配置Application



选择Porject为jldp-web

## 运行Application



执行完成后如下：



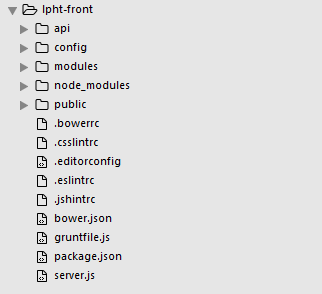
正常启动后，打开浏览器，输入<http://localhost:8080/api/security/logs/queryBaseLogs>，有返回json则表示

# 开发说明

## 前端说明

### 前端目录说明

#### 主体目录说明



api目录 用于存放本地json文件

config目录 前端项目的配置文件存放目录

node\_modules/public目录 依赖包存放目录

modules目录 开发使用的目录（开发人员操作的目录）。

.bowercc:bower配置文件

.csslinttrc/.eslintrc/.jshintrc/editorconfig 规则配置文件

bower.json/package.json:bower/npm依赖包配置文件

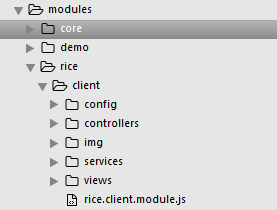
gruntfile.js:grunt配置文件

server.js 前端服务配置文件

#### 模块开发目录说明

如下图所示，modules目录存放对应模块的目录，其中rice目录为开发的具体业务模块目录

在开发过程中，开发人员主要操作controllers、views、config目录



##### config目录

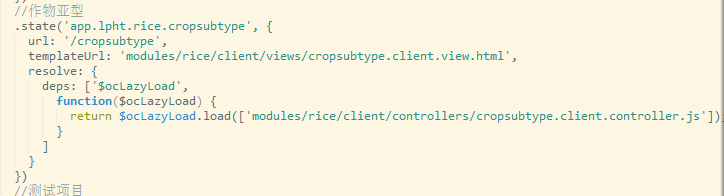
路由等配置文件存放目录



rice.client.routes.js:rice模块的路由文件。

路由文件控制url的跳转。开发新的视图文件和视图控制文件均要在路由文件中配置。

如下图



图中第一行

app.lpht.rice.cropsubtype 表示此路由配置的id。

url：此路由配置的url

templateUrl：配置视图文件

resolve：加载视图控制文件或者其他使用的模块。

##### controller目录

视图控制文件



此为视图控制文件存放目录。

命名规范为

模块控制名.client.controller.js

注意：

1. js文件头需要有 'use strict';
2. js中统一使用单引号
3. if中的相等必须使用===，而不是==。

Controller具体开发示例可以参看datacollection.client.controller.js

##### img

图片存放目录

##### services

服务存放目录

##### Views

视图文件目录，此目录为存放模块视图文件。

文件命名方式：

主视图文件：模块名.client.view.html

模态框文件 : 模块名.modal.client.view.html

视图文件具体开发示例参看datacollection.client.view.html

### 前端开发说明

#### 基本规范

1. 保持代码整洁，注释规范，提高内容可读性。
2. 项目内禁止编写css样式表以及在html document元素中规定style样式。
3. Tab缩进距离设置为两个空格。

#### 命名规范

1. 控制器文件名格式:xxx.client.controller.js，视图模板文件名格式：xxx.client.view.html。
2. Controller类名的各单词首字母大写并且以Ctrl结尾：。

#### 基础控件

1. 按钮统一使用button控件且注明type类型。
2. 需要做验证的表单必须使用form（不能用div代替），否则无法调用angularJs的验证方法。
3. button控件使用小型按钮样式：。
4. input控件使用小型表单样式：。
5. 文本使用小型文本样式：。

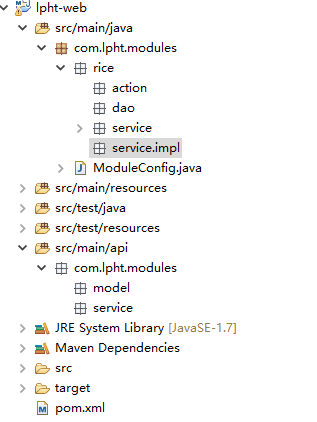
#### UI Grid

1. 每页显示个数可选项参数：。

总页数参数要给初始化数值：。

## 后端说明

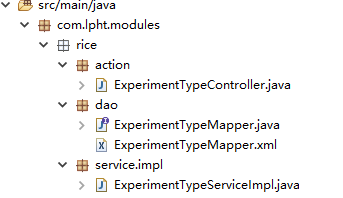
### 后端目录说明



目录说明：

src/main/api 模块对应的model和service接口

src/main/java 模块对应的action，dao，service实现

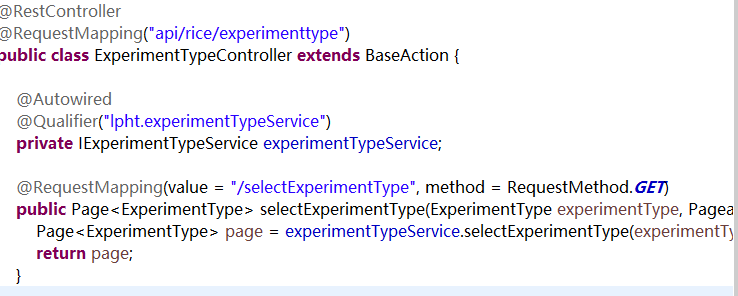


说明：

action/dao/service分别对应各自的实现类

### 后端开发说明

#### Action



说明：

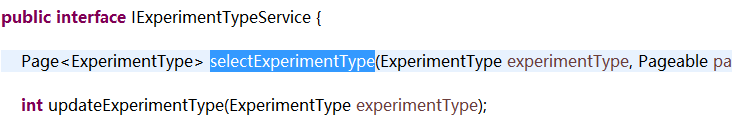
1. 所有的controller类均要继承baseaction
2. 所有的controller类均要加上restcontroller注解以及requestmapping映射注解。
3. 注入的service均使用autowired以及qualifier指定注入service。
4. Action中所有方法的注解均要指定请求方法 get/post

注意：

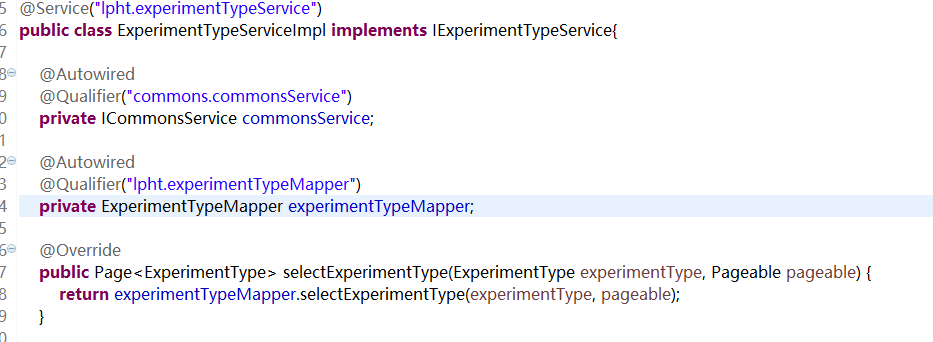
如果action对应方法需要安全验证，那么需要在方法名上加上@Security注解。

#### Service

##### 接口



##### 实现

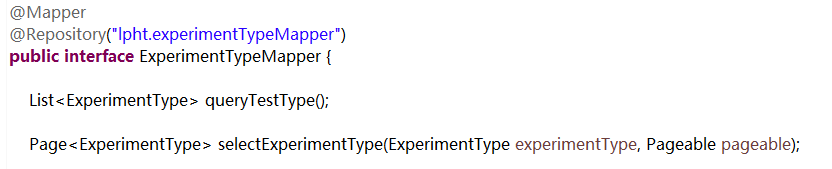


说明：

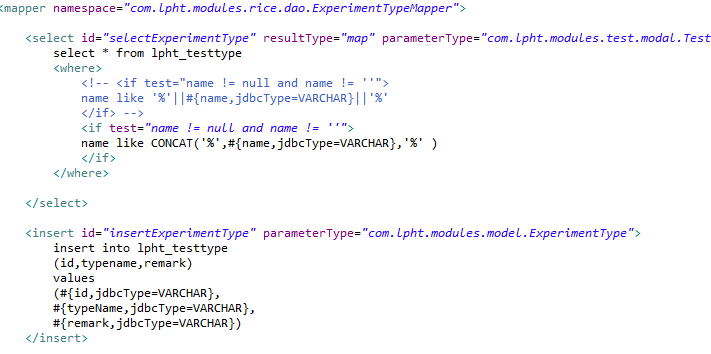
1. 在实现类上加上service注解以及对应名称。
2. Commonnservice为公用service。可以通过此service获取uuid。
3. Experimenttypemapper为dao接口
4. 接口的注入使用autowired和qualifier指定注入类

#### DAO

##### 接口



##### XML映射



说明：

1. 映射文件的命名空间与接口文件路径一致。
2. Select/update/del/insert 标签的id均与接口文件中的方法名称一致。