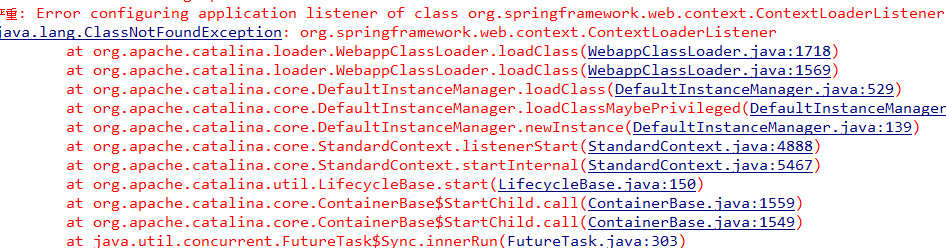
项目第三天

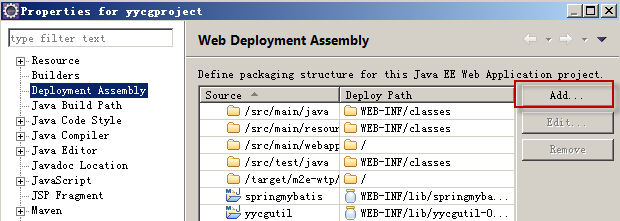
# 复习：

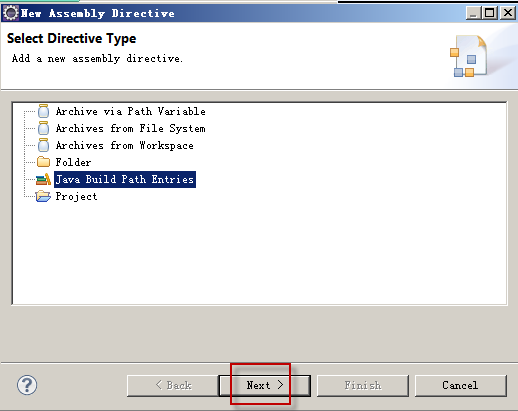
## 问题

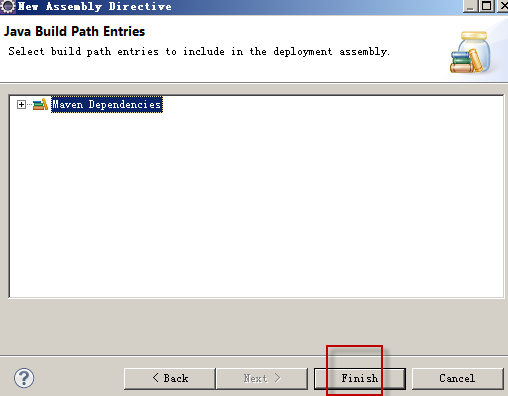
Maven工程部署到tomcat目录，所依赖的jar无法部署问题：

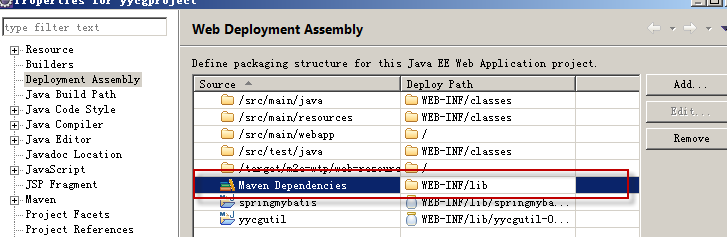


通过查看tomcat中工程目录 ,WEB-INF/lib下没有jar









上图中出现maven依赖的jar部署到WEB-INF/lib。

重新发布工程到tomcat，观察工程目录WEB-INF/lib，已经将工程依赖的jar部署到tomcat。

## 课程复习

用户管理数据模型：

学习表的方法：

表的作用，表中记录了什么业务数据？

数据库级别表与表之间的关系（外键关系）

表的业务意义，以及表与表之间**业务关系**（一对一、一对多、多对多。。）。

单位信息表：

监督单位表：userjd

记录了卫生局和卫生院信息

供货商表：usergys

记录了供货商信息

供货区域表：usergysarea

记录了供货商所供货区域信息

医院表：useryy

记录了医院的信息

注意：系统中业务表和单位表关联，比如采购单信息表记录单位id，记录单位所创建的采购单。

系统用户表：sysuser

记录了系统中所有的登陆用户信息

行政区域表：bss\_sys\_area

记录了系统中用到行政区域

数据库级别的关系：

Userjd表中有外键bss\_sys\_area（areaid）

useryy表中有外键bss\_sys\_area（areaid）

usergysarea表中有外键bss\_sys\_area（areaid）、和usergys(id)

业务关系：

在sysuser表中加了sysid（单位id），根据用户类型（groupid），决定sysid存储的内容，如果groupid为1或2sysid等于Userjd表中的主键，如果groupid为3，sysid等于Useryy表中的主键

用户管理查询开发方法

本系统统一的开发查询列表方法：

**第一步：实现dao**

自定义mapper.xml(映射文件)、mapper.java（接口）

在mapper.xml中定义statement：

1 根据查询条件查询列表的总数，返回int

2 根据查询条件查询列表

resultType=custom扩展类

parameterType=包装类

**第二步：实现service**

定义两个方法：

查询列表总数

查询列表

Service接口定义时：

确定service接口功能

确定service接口参数，业务参数尽量细化，尽量简单类型

Service接口实现时：

根据需求分析中约束条件校验（业务逻辑），如果校验不通过抛出系统自定义的系统

调用dao层对数据时行持久化。

**第三步：实现action**

开发两个方法

查询列表页面方法，返回string（jsp）

查询列表json结果集方法，使用@responseBody注解返回DataGridResultInfo类型。

**第四步：实现页面**

先实现静态页面布局

添加动态标签：

使用jquery easyui的datagrid展示查询列表，支持分页。

Datagrid定义：

定义列

定义工具栏

加载datagrid

用户添加方法

* Dao：

使用自动生成的mapper进行数据添加、删除、更新操作

* service：

service接口功能：添加一个新用户

service接口参数：用户信息（使用XXXCustom扩展类）

service实现：

根据需求分析中约束条件校验（业务逻辑），如果校验不通过抛出系统自定义的系统

调用dao层对数据时行持久化。

* action：

提交方法

系统中所有的提交方法(所有post操作)，统一返回SubmitResultInfo提交结果类型，使用responseBody转json

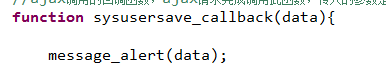
* 页面：

使用jquery 提供ajax的form提交组件进行数据提交。



定义提交回调函数：

使用message\_alert解析提交结果（就是SubmitResultInfo的json数据）



## 开发规范

custom扩展类：自定义pojo作为扩展类，继承查询列较多原始po类。

用于，查询结果显示、表单数据显示、存储查询条件

命名规则：XXXXCustom.java

包装类：自定义一个pojo，将custom扩展类、po类包含进来

用于，提交表单向action传递数据，通过包装类从页面传递数据层（mybatis）

命名规则：XXXXXQueryVo.java

action数据查询列表json结果集统一使用DatagridResultInfo类返回。

action提交方法形参统一使用包装类。

action提交方法结果统一使用SubmitResultInfo类返回。

# 教学计划

实现国际化配置

工具类的使用（掌握）

按照系统开发规范实现用户修改、用户删除

数据字典

实际操作：

根据需求实现功能，老师提出需求，学生实现功能

# 实现国际化配置

## 需求：

在service代码中将给用户提示的信息硬编码了，不利于系统维护。



## 实现：

Java国际化主要通过如下3个类完成

java.util.ResourceBundle:用于加载一个资源(配置文件)

java.util.Locale：对应一个特定的国家/区域、语言环境。

java.text.MessageFormat：用于将消息格式化

//根据操作系统环境获取语言环境

Locale locale = Locale.*getDefault*();

System.*out*.println(locale.getCountry());//输出国家代码

System.*out*.println(locale.getLanguage());//输出语言代码s

//加载国际化资源（classpath下resources目录下的messages.properties，如果是中文环境会优先找messages\_zh\_CN.properties）

ResourceBundle rb = ResourceBundle.*getBundle*("resources.messages", locale);

String retValue = rb.getString("102");//101是messages.properties文件中的key

System.*out*.println(retValue);

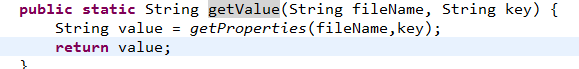
//信息格式化，如果资源中有{}的参数则需要使用MessageFormat格式化，Object[]为传递的参数，数量根据资源文件中的{}个数决定

String value = MessageFormat.*format*(retValue, **new** Object[]{100,200});

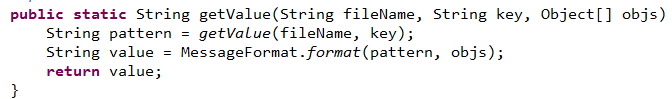
System.*out*.println(value);

使用工具类ResourcesUtil，从资源文件中取配置信息：

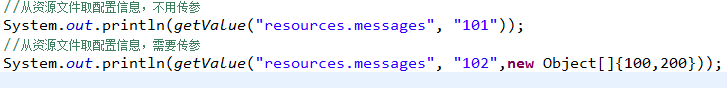
不用传参取配置信息：



需要传参数取配置信息：



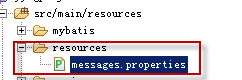
工具类测试:

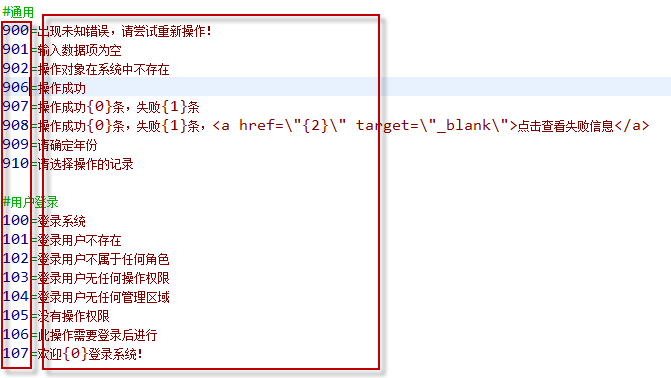


## 本系统国际化配置

定义资源文件

将系统中所有的提示信息（操作成功信息、操作异常信息等）统一配置在资源文件，在java代码只需要调用工具类从资源中文件取出配置信息，避免硬编码.





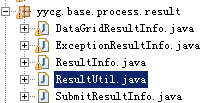
国际化资源文件使用

使用工具类从资源文件中获取配置信息。



实际使用时将从资源文件中取出的配置信息填充到ResultInfo中。

提供封装类去构造ResultInfo，从资源文件中取配置信息。

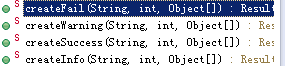


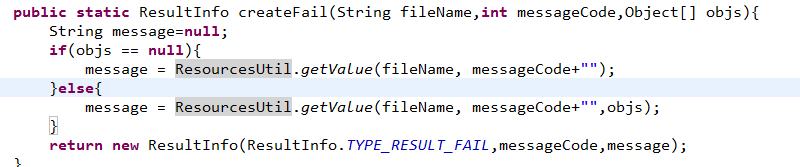
ResultUtil提供构造ResultInfo的方法包括：

构造一个成功结果：createSuccess

构造一个失败结果：createFail

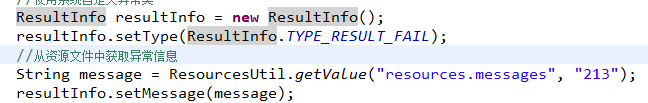
…





createFail方法如何使用？

将以下代码使用ResultUtil类实现.



封装后：



将下边的代码封装：



封装后：

使用ResultUtil类实现

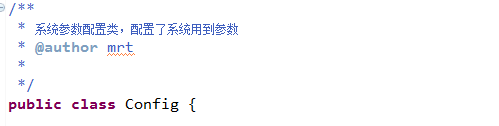


提交结果封装：



## 系统config

将系统中配置信息统一通过静态变量配置此类中。



使用方法，比如：



# 用户删除

## 需求

用户出现变更、用户不存在了，将系统用户从系统中删除。

前置条件（约束）：

用户必须存在方可删除

后置条件（数据库操作）

从sysuser表删除用户信息

## 实现

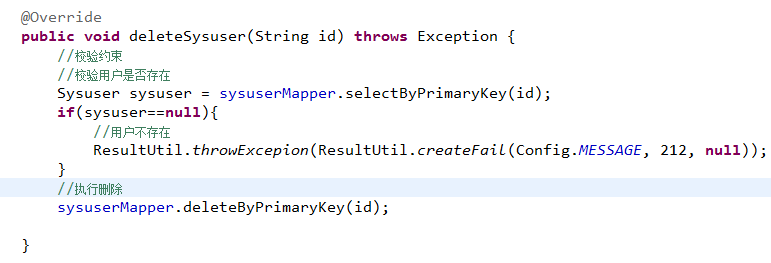
dao

使用自动生成 的mapper从sysuser表删除一个用户

service

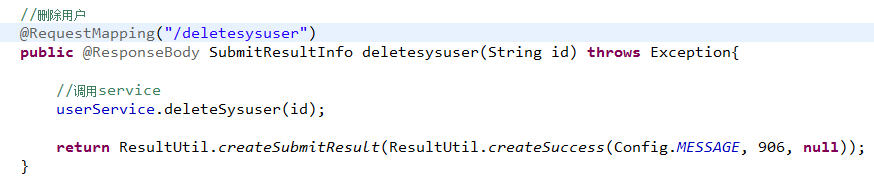
接口功能：将系统用户从系统中删除

接口参数：用户id



action

实现用户删除方法(提交方法)。

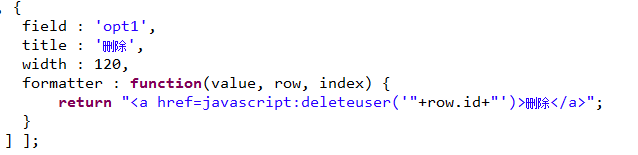


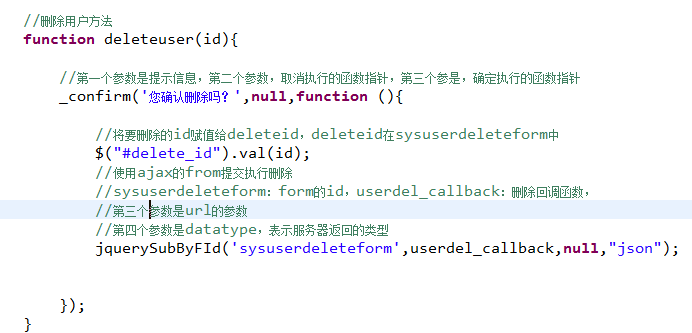
页面

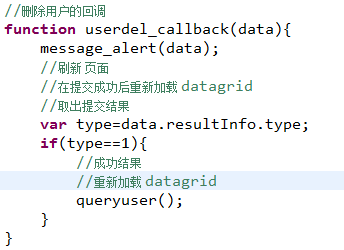
通常情况，针对单条记录操作，建议在记录上添加链接操作。

修改queryuser.jsp：

添加删除链接



定义删除方法  




删除操作使用ajax的form提交 ，需要在页面中定义单独 删除form，form中有一个删除用户id。



# 用户修改

## 需求 ：

修改用户的信息，可以修改用户账号、用户的名称、用户密码、用户类型、用户所属单位、用户状态。

前置条件（约束）：

修改用户账号不允许暂用别人的账号

用户变更单位必须是存在的

后置条件（数据库操作）

从sysuser表修改用户信息

## 实现 ：

dao

使用自动生成 的mapper从sysuser表修改一个用户

service

接口功能：修改用户信息

接口参数：修改用户的id、修改用户的信息

约束分析：修改用户账号不允许暂用别人的账号

如果用户的账号不修改不用校验账号是否暂用别人的账号

如果用户修改账号了，才需要校验账号是否暂用别人的账号

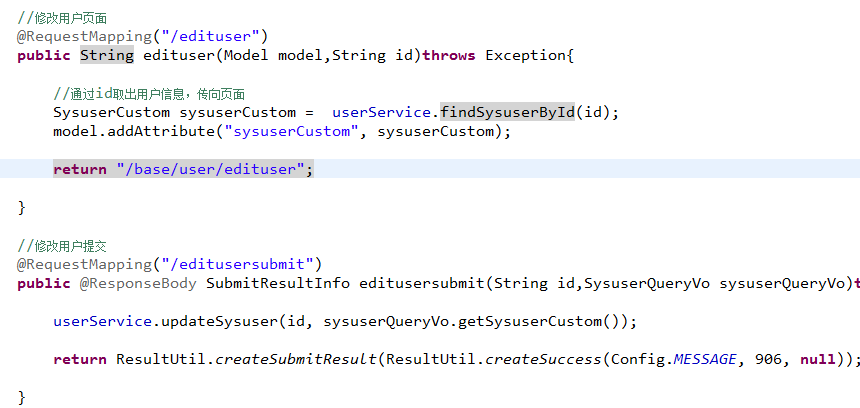
通过修改的账号查询sysuser系统用户表，如果查询到说明暂用别人的账号

根据用户id获取用户信息：



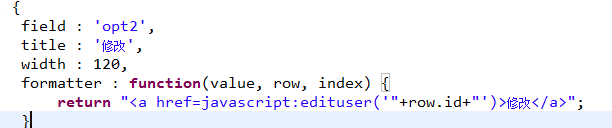
action

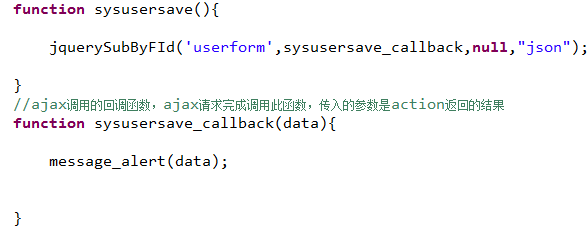
两个方法：



页面

在queryuser.jsp添加修改链接。



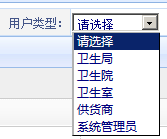


# 数据字典

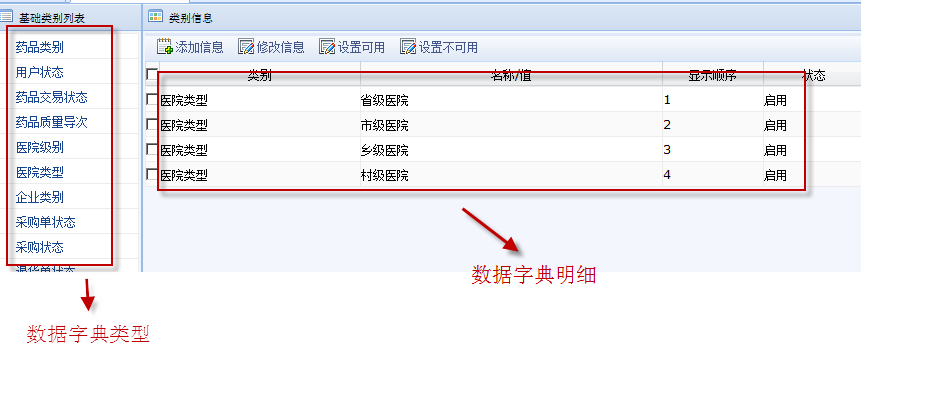
## 什么是数据字典

将如下这些具有相同类型的配置项，配置到系统的**数据字典表**中，方便系统维护，由超级管理员统一在后台进行数据字典维护，如果用户需求要增加变更配置项，只需要修改数据字典表记录即可，不需要修改代码。





系统管理员登陆系统后台，进行数据字典维护，



## 数据字典需求

相同类型的配置项：

在系统中创建一个张记录数据**字典类型**：

数据字典类型表

创建一张表记录数据**字典明细**：

数据字典明细表



将上边变化灵活的配置项叫做：“**普通配置项**”

将上边固定的配置项（**每个配置顶都有一个代码**）：“**业务代码**”

## 数据字典表结构

字典类型表DICTTYPE：

记录数据字典类型

字典明细表DICTINFO：

记录数据字典明细

外键：字典类型id

TYPECODE

字典明细表DICTINFO

ID：主键

TYPECODE：外键

DICTCODE：业务代码

INFO：配置项名称

字典类型表DICTTYPE

TYPECODE：类型id主键

TYPENAME：类型名称

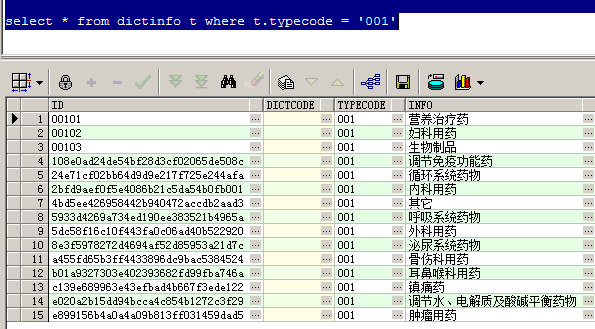
字典明表中如何存储普通配置项和业务代码。

普通配置项存储：

普通配置项名称存储在DICTINFO表中info字段

普通配置项对应的类型id存储在DICTINFO表中TYPECODE

查询普通配置顶：



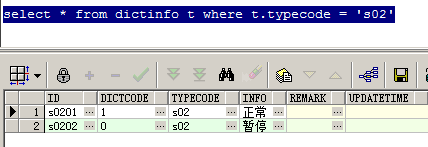
业务代码存储：

业务代码对应的名称存储在DICTINFO表中info字段

业务代码对应的类型id存储在DICTINFO表中TYPECODE

**业务代码存储在DICTINFO表中DICTCODE(是和普通配置顶的区别)**

业务代码查询：



## 数据字典使用

1、在页面上的下拉框中显示业务代码或普通配置

比如：在系统用户查询页面，查询条件就是用户类型，用户类型不能在jsp上硬编码，需要在action方法中取出用户类型所有配置项，在jsp页面动态遍历。

2、在查询业务表需要关联查询出业务代码对应的名称

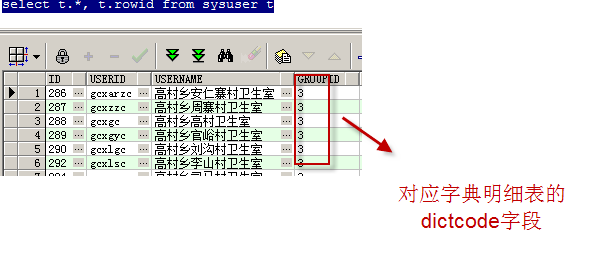
在查询业务表时，根据业务代码，关联查询出代码对应的名称。

关联查询字典明细时**指定业务代码和类型id**



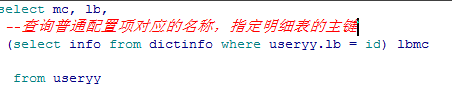
如何存储?

在业务表（比如系统用户表）中，对于业务代码将明细表中的dictcode字段值存储在业务表中。



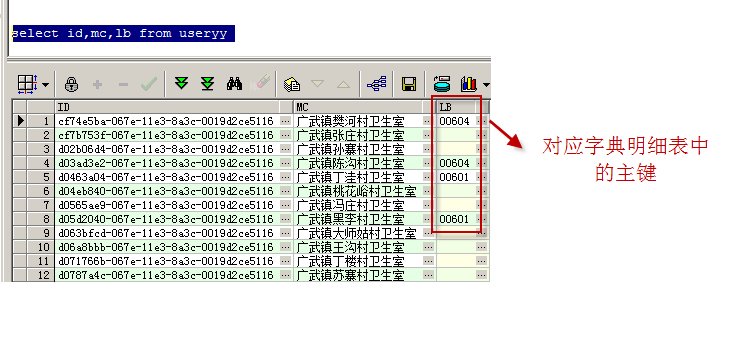
3、在查询业务表需要关联查询出普通配置项所对应的名称

关联查询字典明细时**指定明细表的主键**



如何存储?

在业务表（比如医院表）中，对于普通配置项将明细表中的id字段值存储在业务表中。



## 小结

普通配置项：对业务数据进行简单的归类，这些归类受用户要求变化较灵活，将这些配置项作普通配置项配置数据字典表。

业务代码：系统运行所必须的，在系统设计时定义的固定代码，这些代码可能需要在程序代码进行硬编码。

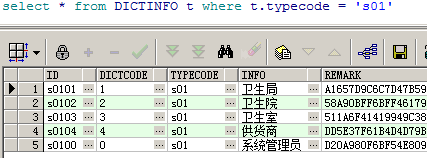
## 数据字典在系统中应用

用户查询页面用户类型下拉框

用户类型：属于业务代码

获取用户类型下所有明细：

从数据字典明细表查询，根据typecode查询



如何实现：

定义一个service接口：

接口功能：根据字典类型查询字典明细

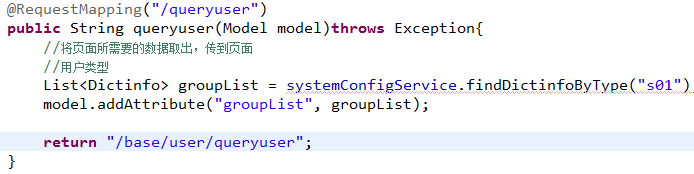
接口参数：typecode（字典类型id）





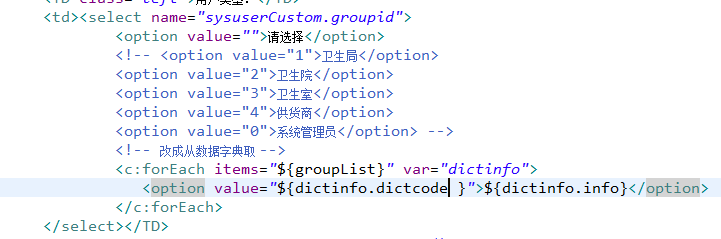
Action：

修改用户查询页面方法，调用systemConfigService查询用户类型信息，将用户类型列表信息传入页面。



页面：

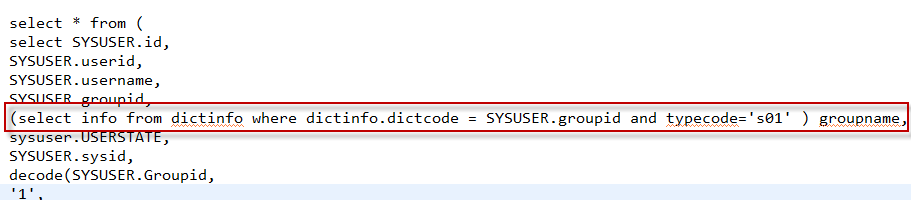
将用户类型下拉框值，改成遍历groupList列表。



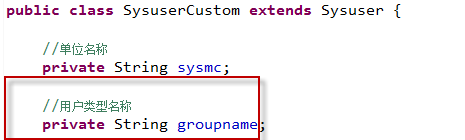
用户查询列表中用户类型列

Dao:

修改用户查询列表mapper，添加一列需要关联用户类型的代码查询代码对应的名称。

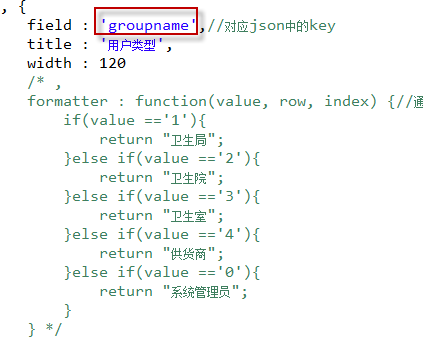
**

*修改mapper的resultType类型，添加groupname属性。*

**

页面：

修改datagrid的列定义，



普通配置项下拉框

在action中使用



查询出普通配置项列表，

页面中下拉框遍历list（option的value是${dictinfo.id}）

<c:forEach items=*"*${list}*"* var=*"dictinfo"*>

<option value=*"*${dictinfo.id }*"*>${dictinfo.info}</option>

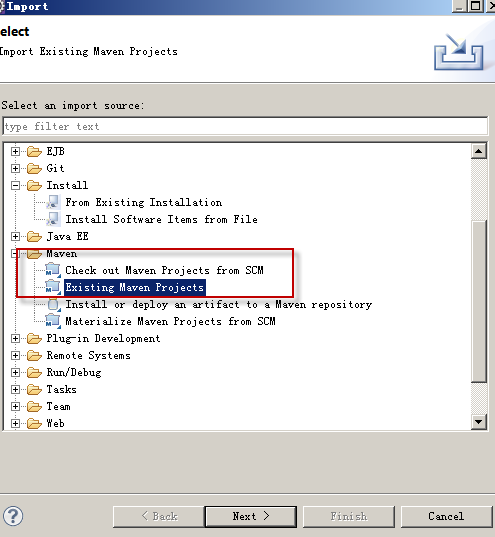
</c:forEach>

# 推荐学习方法

在eclipse中新创建一个工作区（对应磁盘目录 ）。

按照“开发环境配置.docx”配置jdk环境，maven环境

将老师的所有工程导入到新建工作区中。





运行工程，调试工程，正常运行。

按照老师说的需求去实现。

需求1：

用户查询列表，添加查询条件：用户状态(从数据字典中取)

用户添加、修改表单添加：联系方式、电子邮件、地址、电话。