**IUAP平台使用指南**

**开发环境安装**

用友网络科技股份有限公司

二○一八年五月

**此文档为保密文档，未经用友软件股份有限公司书面同意，不得向任何单位或个人提供、转让本文档中的任何内容，用友软件股份有限公司将保留对泄漏文档内容的起诉权利。**

**IUAP平台使用指南**

|  |  |  |
| --- | --- | --- |
| 文件状态：  [ ] 草稿  [**√**]正在修改  [] 正式发布 | 文件标识： |  |
| 当前版本： |  |
| 作者： |  |
| 完成日期： |  |

版本历史

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本 | 作者 | 参与者 | 起止日期 | 备注 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

审核记录

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本 | 姓名 | 职位 | 审阅日期 | 备注 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

目 录

[第一章 概述 1](#_Toc515710200)

[1.1 说明 1](#_Toc515710201)

[1.2 环境要求 1](#_Toc515710202)

[第二章 初始化数据库 2](#_Toc515710203)

[第三章 安装反向代理服务器 3](#_Toc515710204)

[3.1 设置 3](#_Toc515710205)

[3.2 启动 3](#_Toc515710206)

[3.3 验证 4](#_Toc515710207)

[第四章 IDE设置 5](#_Toc515710208)

[4.1 启动IDE 5](#_Toc515710209)

[4.2 设置开发环境 5](#_Toc515710210)

[4.2.1 Java设置 5](#_Toc515710211)

[4.2.2 Tomcat设置 6](#_Toc515710212)

[4.2.3 Maven设置 9](#_Toc515710213)

[4.3 导入示例工程 9](#_Toc515710214)

[4.3.1 导入工程 9](#_Toc515710215)

[4.3.2 参数修改 11](#_Toc515710216)

[4.3.2.1 application.properties 11](#_Toc515710217)

[4.3.2.2 applicationContext-persistence.xml 12](#_Toc515710218)

[4.3.3 设置 12](#_Toc515710219)

[4.4 启动tomcat 14](#_Toc515710220)

[4.4.1 部署 14](#_Toc515710221)

[4.4.2 启动 15](#_Toc515710222)

[4.5 验证 15](#_Toc515710223)

[4.5.1 设置 15](#_Toc515710224)

[4.5.2 界面验证 16](#_Toc515710225)

[4.5.3 断点 16](#_Toc515710226)

[第五章 代码结构说明 17](#_Toc515710227)

[5.1 示例工程结构 17](#_Toc515710228)

[5.2 Mybatis介绍 19](#_Toc515710229)

[5.2.1 Mybatis持久化配置 20](#_Toc515710230)

[5.3 项目配置解析 21](#_Toc515710231)

[5.3.1 Spring集成 21](#_Toc515710232)

[5.3.2 SpringMVC 22](#_Toc515710233)

[5.3.2.1 配置解析 22](#_Toc515710234)

[5.3.2.2 SpringMVC常用注解 22](#_Toc515710235)

[5.4 后台代码解析 23](#_Toc515710236)

[5.4.1 配置 24](#_Toc515710237)

[5.5 前台代码解析 24](#_Toc515710238)

[5.5.1 前端页面架构 24](#_Toc515710239)

[5.5.2 单页面应用(SPA) 25](#_Toc515710240)

[5.5.3 前端路由 25](#_Toc515710241)

[5.5.4 AMD模块化规范 26](#_Toc515710242)

[5.6 代码规范说明 27](#_Toc515710243)

[5.6.1 后台代码分包 27](#_Toc515710244)

[5.6.2 包名建议规范 27](#_Toc515710245)

[5.6.3 Java代码建议规范 27](#_Toc515710246)

[5.6.4 前台包名建议规范 28](#_Toc515710247)

[5.6.5 前台代码建议规范 28](#_Toc515710248)

[第六章 额外配置 29](#_Toc515710249)

[6.1 修改Nginx配置 29](#_Toc515710250)

# 概述

## 说明

1. 开发环境是每个开发人员在各自的设备上安装
2. 安装工具主要包括反向代理服务器nginx、应用服务器tomcat和开发工具eclipse、管理工具maven

## 环境要求

1. JDK 推荐使用1.8及以上（例如：jdk1.8.0\_25\_x64）
2. IEDeclipse、IntelliJ IDEA。例如：（Eclipse IDE for Java EE Developers （Windows下64位，luna sr2）/ IntelliJ IDEA 2016.3.1）
3. 应用服务器采用tomcat8（例如：apache-tomcat-8.0.51及以上）
4. maven使用安装盘的版本（Maven3.2.2）

备注：安装盘中都进行了统一提供。

# 初始化数据库

在iuap-example文件夹下有示例代码的数据库初始化脚本，请按照数据库类型进行初始化。

1. 执行“iuap-example\菜单初始化”下面对应数据的sql语句，对应用平台的菜单进行初始化。
2. 执行“iuap-example\样例建表语句”下面对应数据的sql语句，对开发环境工程中样例的数据库建表语句。

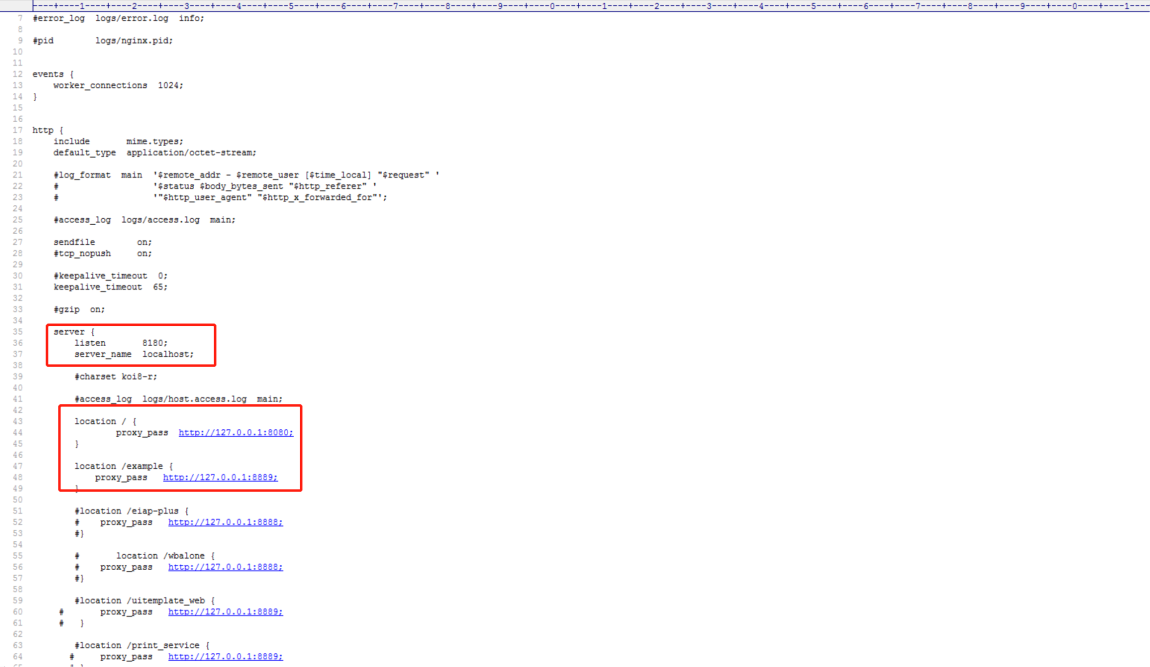
# 安装反向代理服务器

反向代理服务器采用nginx（nginx-1.13.1）

## 设置

需要对代理进行设置，nginx分别对应用平台和开发环境的tomcat进行代理设置。

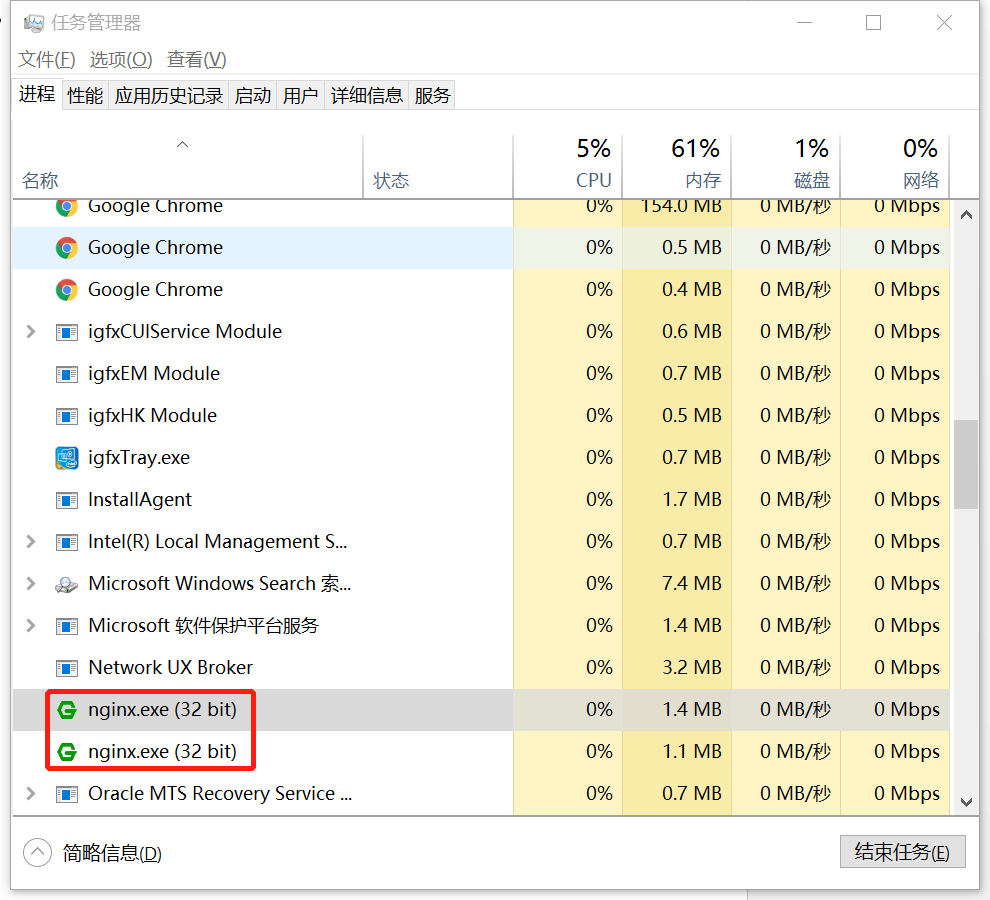
默认提供的nginx中，已经按照预制好的端口号进行了设置。



1. nginx设置

## 启动

通过nginx.exe启动，启动时一闪而过，在任务管理器中查看，如果有此进程则说明已经启动成功。



1. 启动nginx

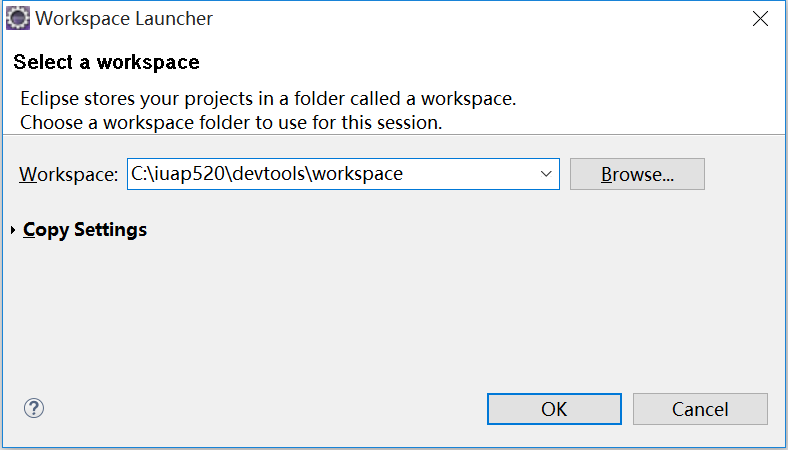
## 验证

通过访问http://127.0.0.1:8180/wbalone/进行验证，如果能够正常访问说明代理成功。

# IDE设置

## 启动IDE

选择一个工作空间

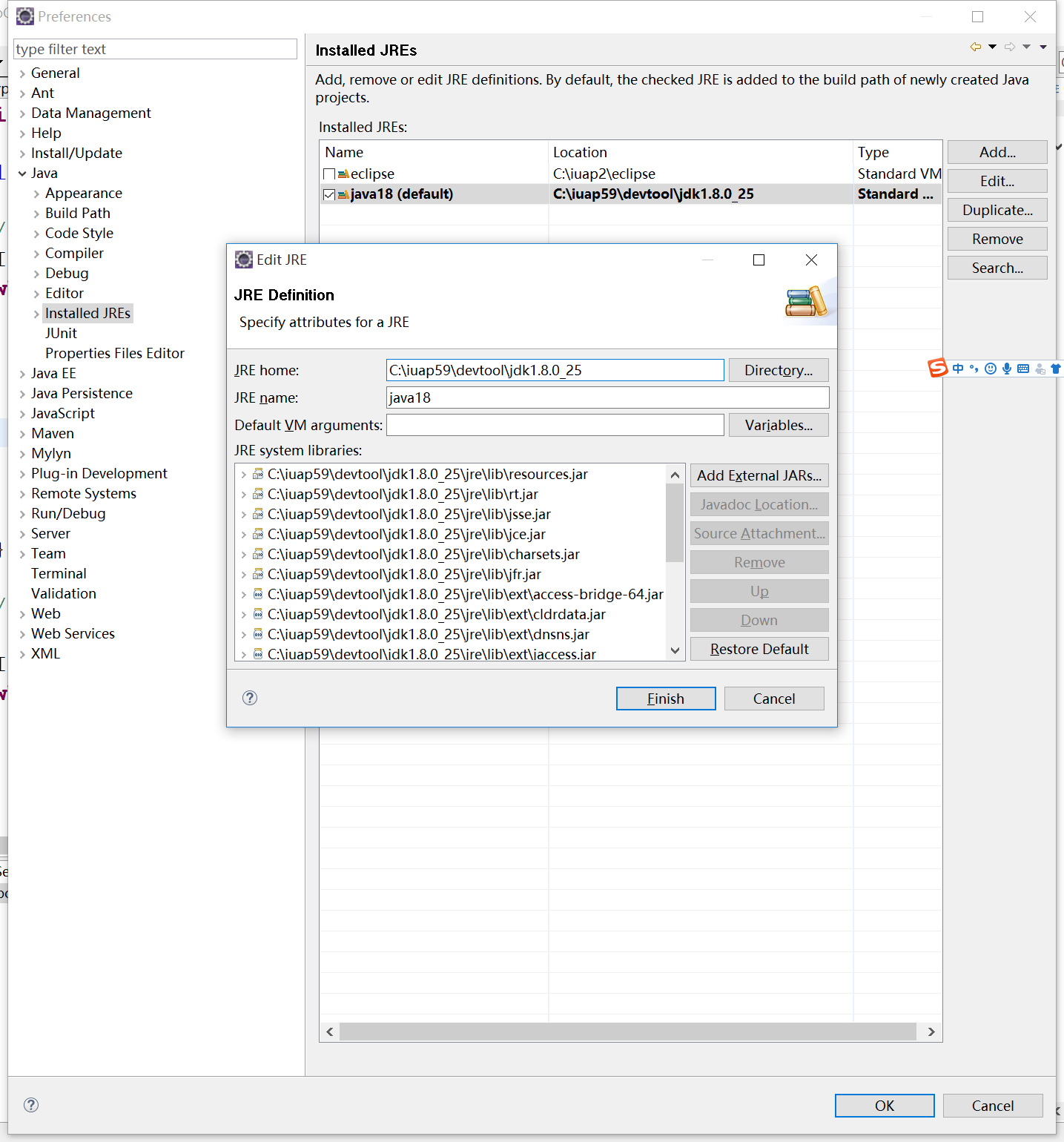


1. 选择工作空间

## 设置开发环境

### Java设置

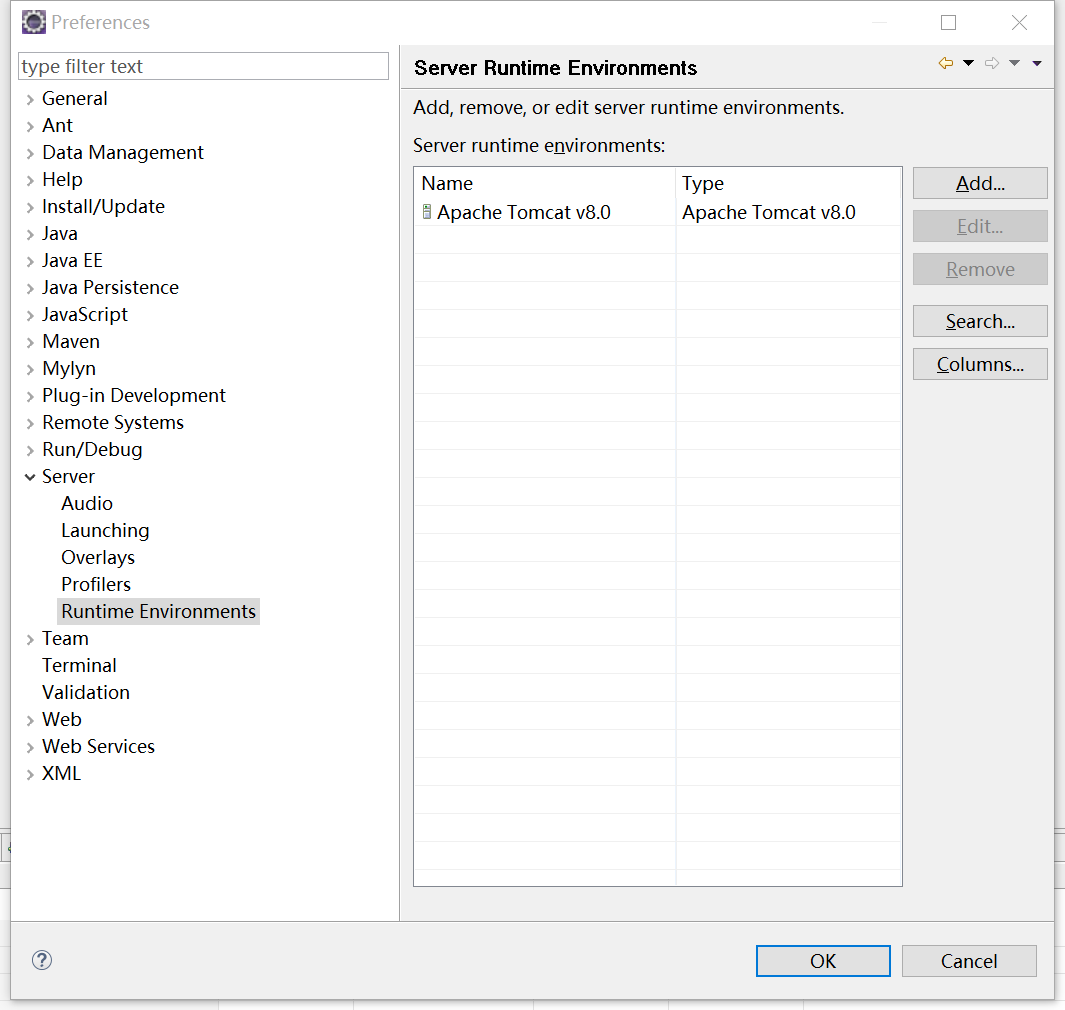
此处为标准设置，选择在安装工具包中提供的java环境。



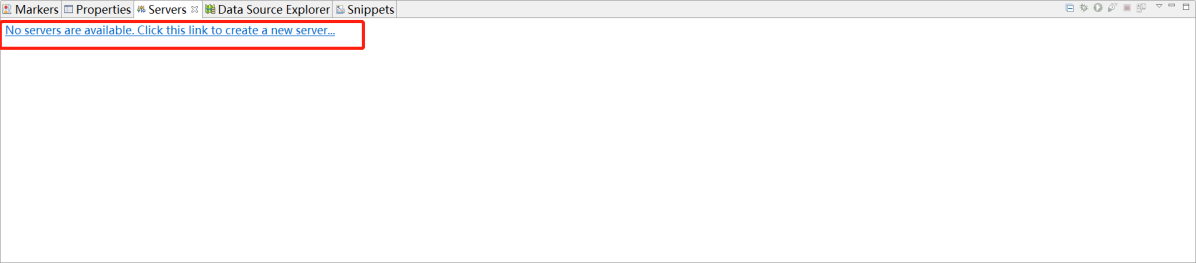
1. 设置java

### Tomcat设置

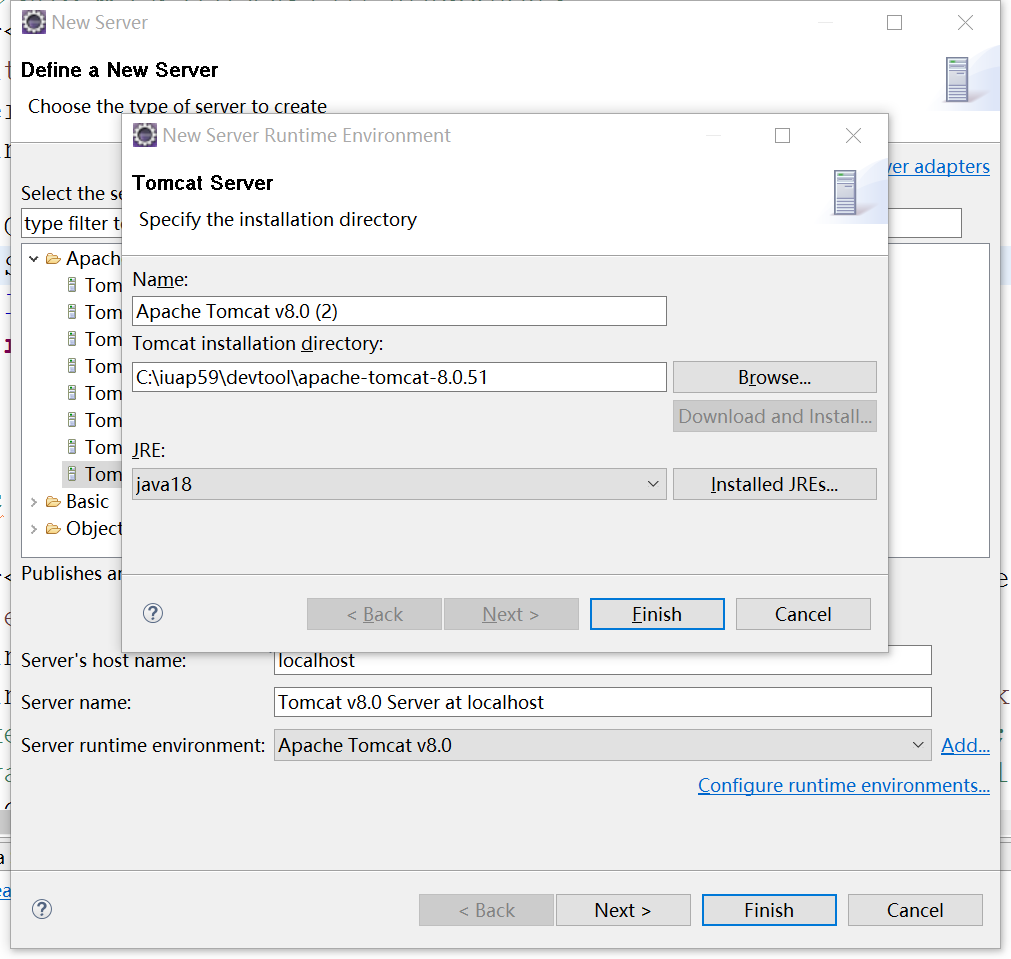
此处为标准设置，选择在安装工具包中提供的tomcat环境。



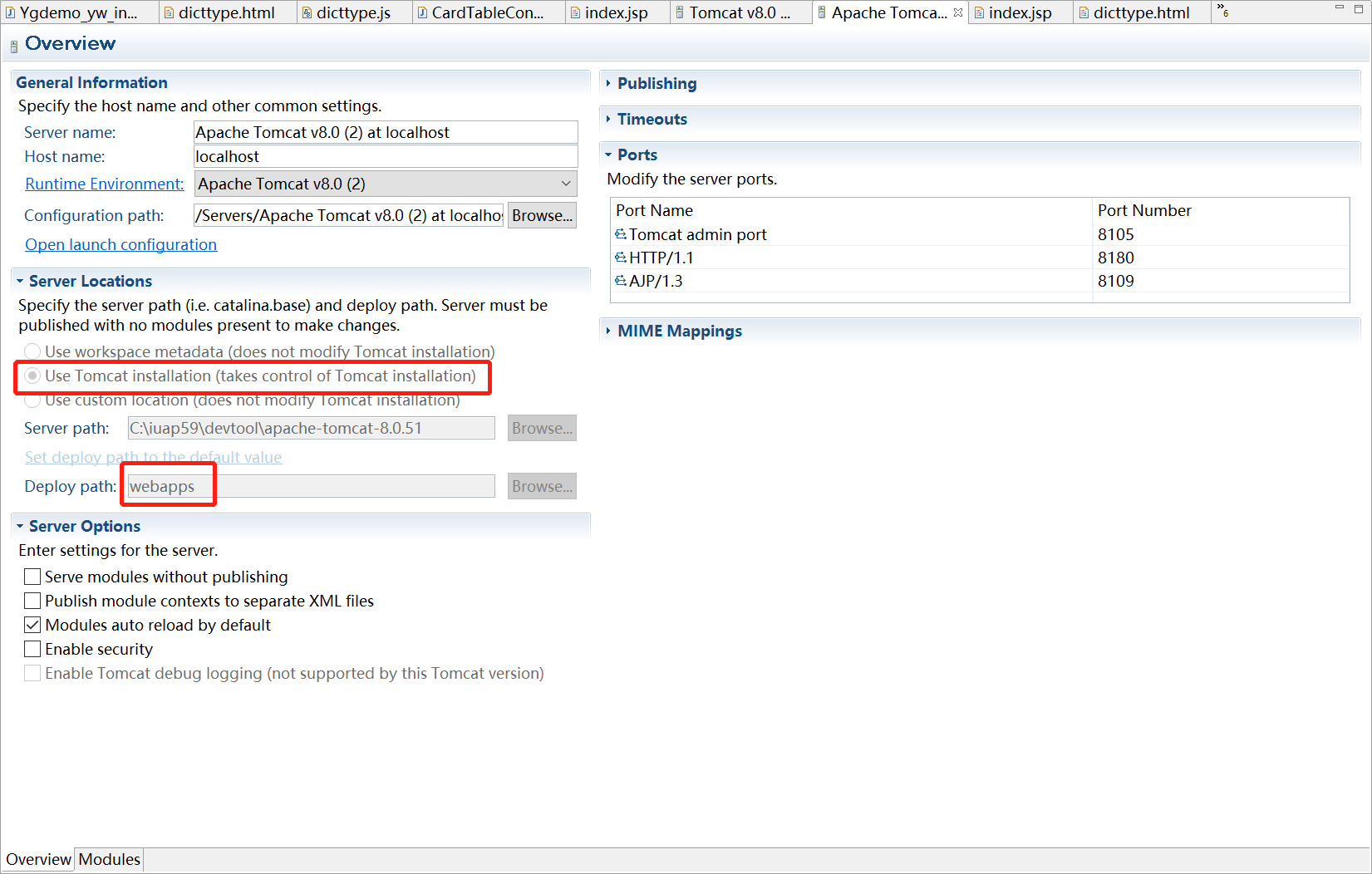
1. 添加tomcat运行时



1. 添加tomcat服务



1. 添加tomcat服务



1. Tomcat设置

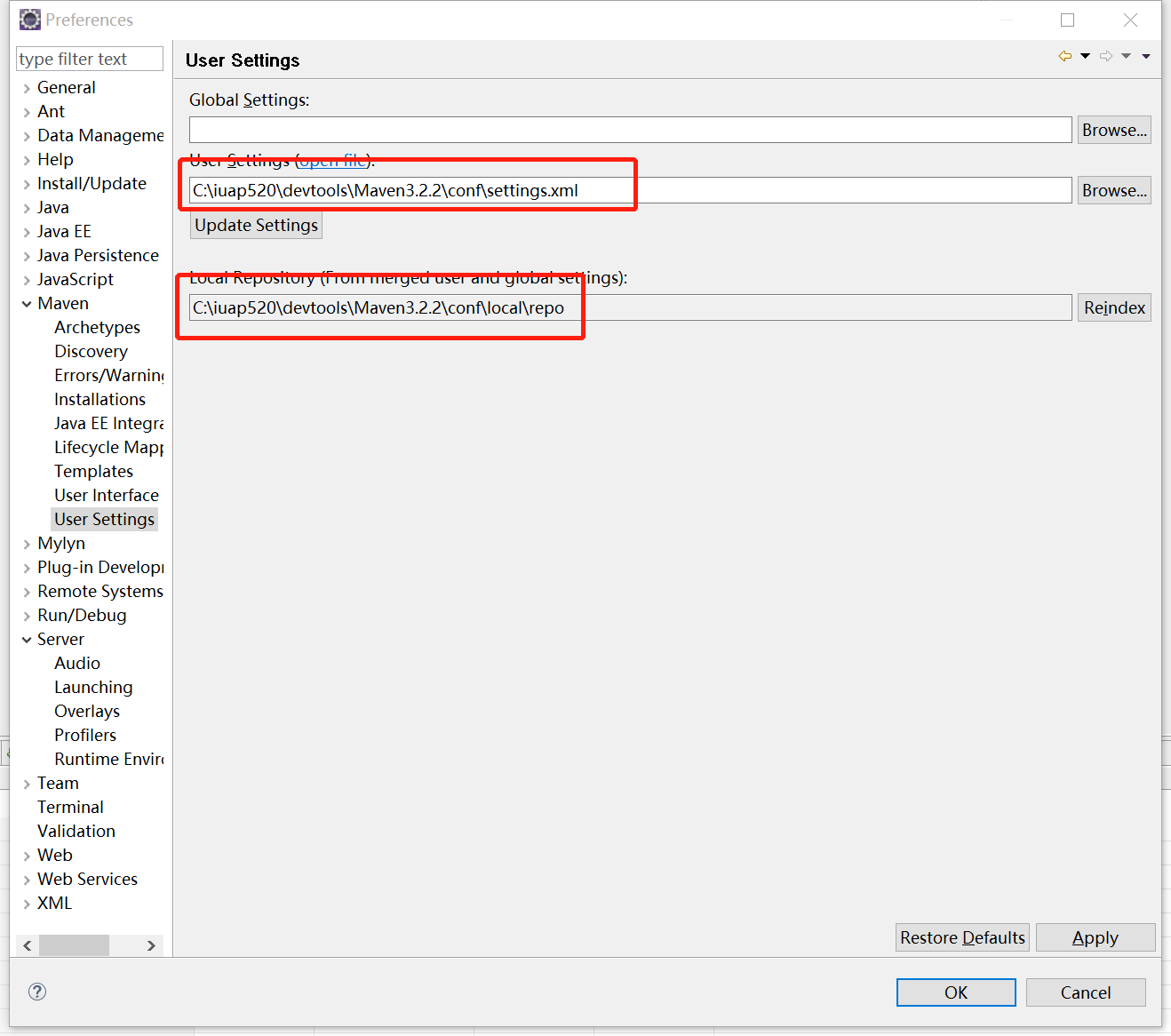
### Maven设置

1. 修改maven设置

修改Maven3.2.2\conf\settings.xml文件中localRepository的设置，一般改为当前目录下的local\repo文件夹。



2、修改eclipse中设置，此处为标准设置

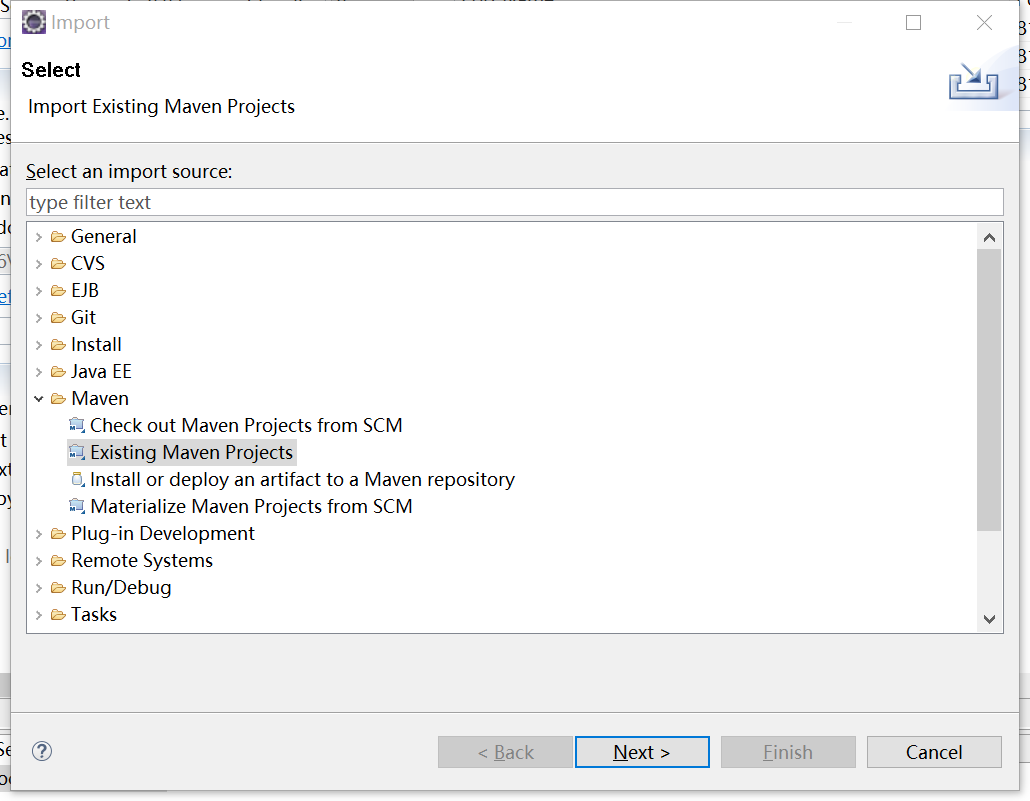


1. 设置maven

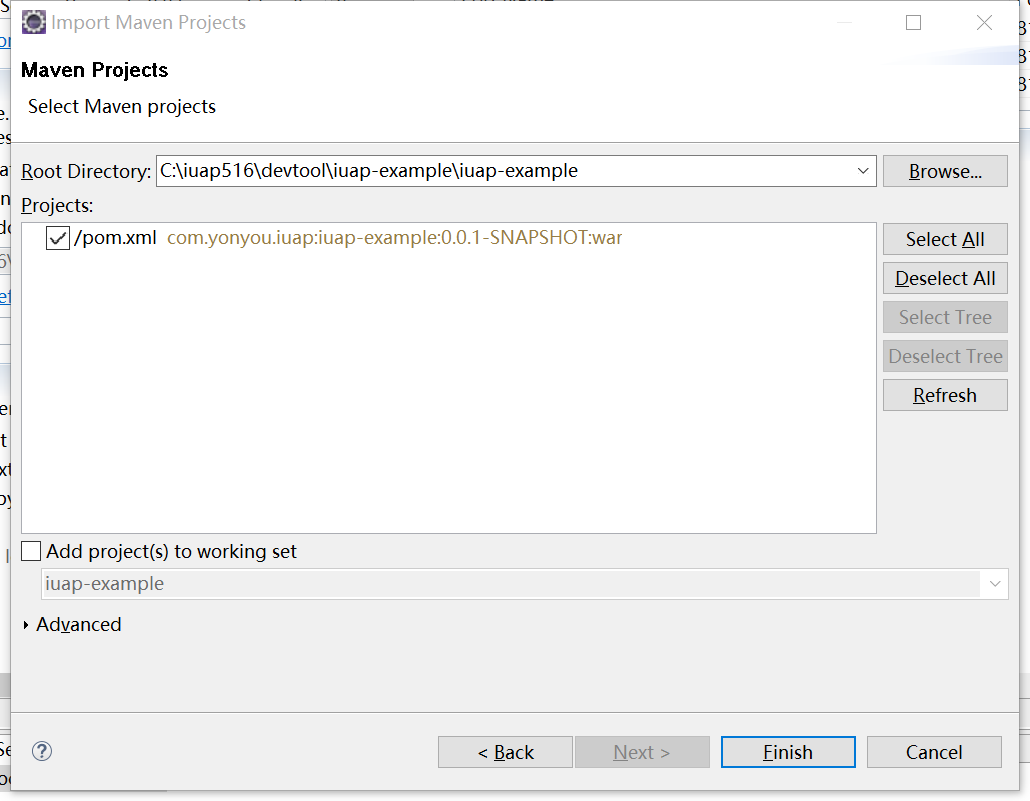
## 导入示例工程

### 导入工程

按照标准的maven工程进行导入



1. 导入示例工程

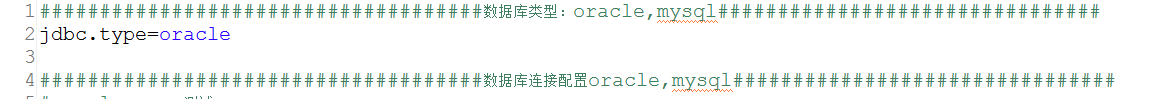


1. 导入示例工程

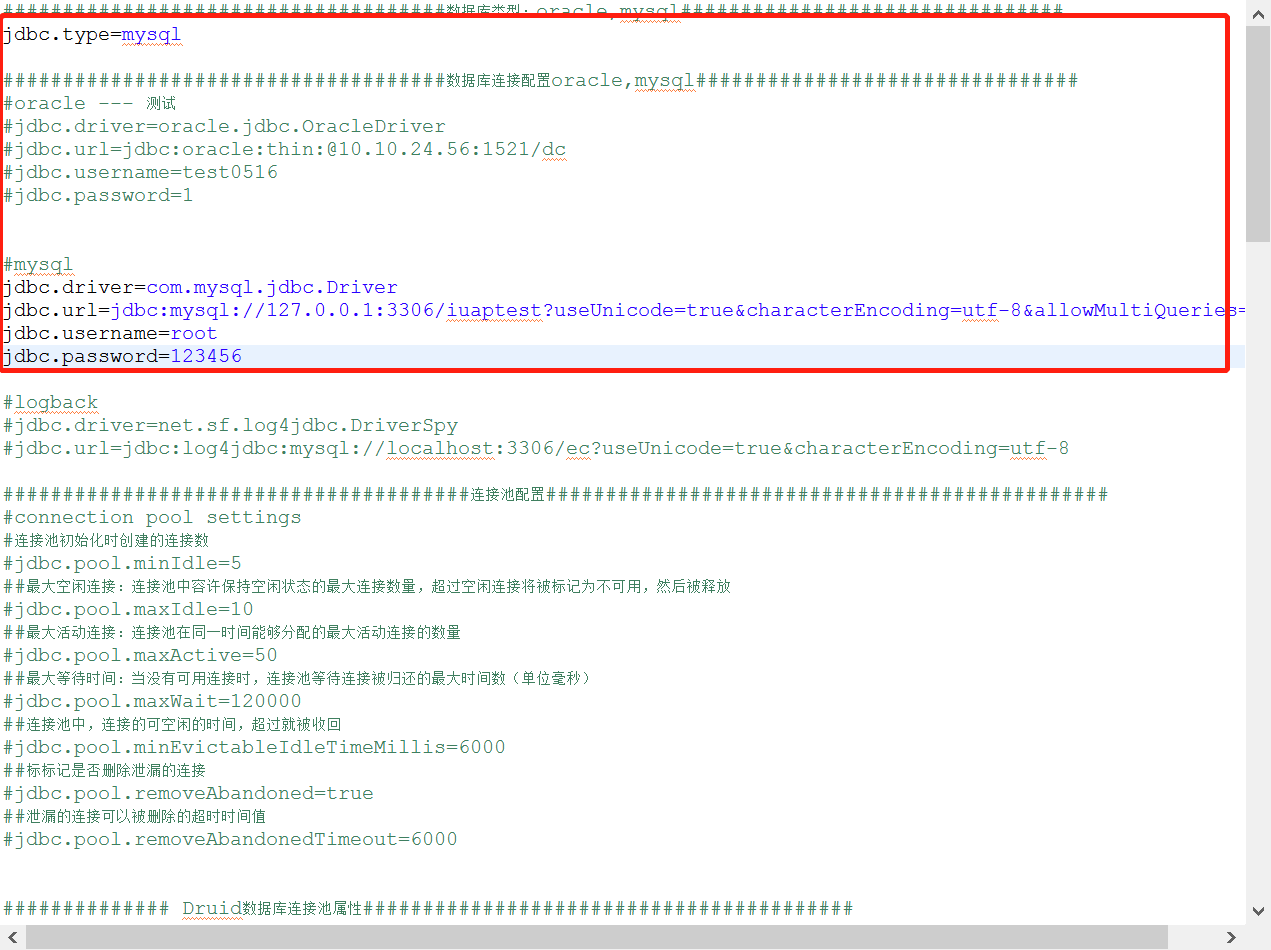
### 参数修改

#### application.properties

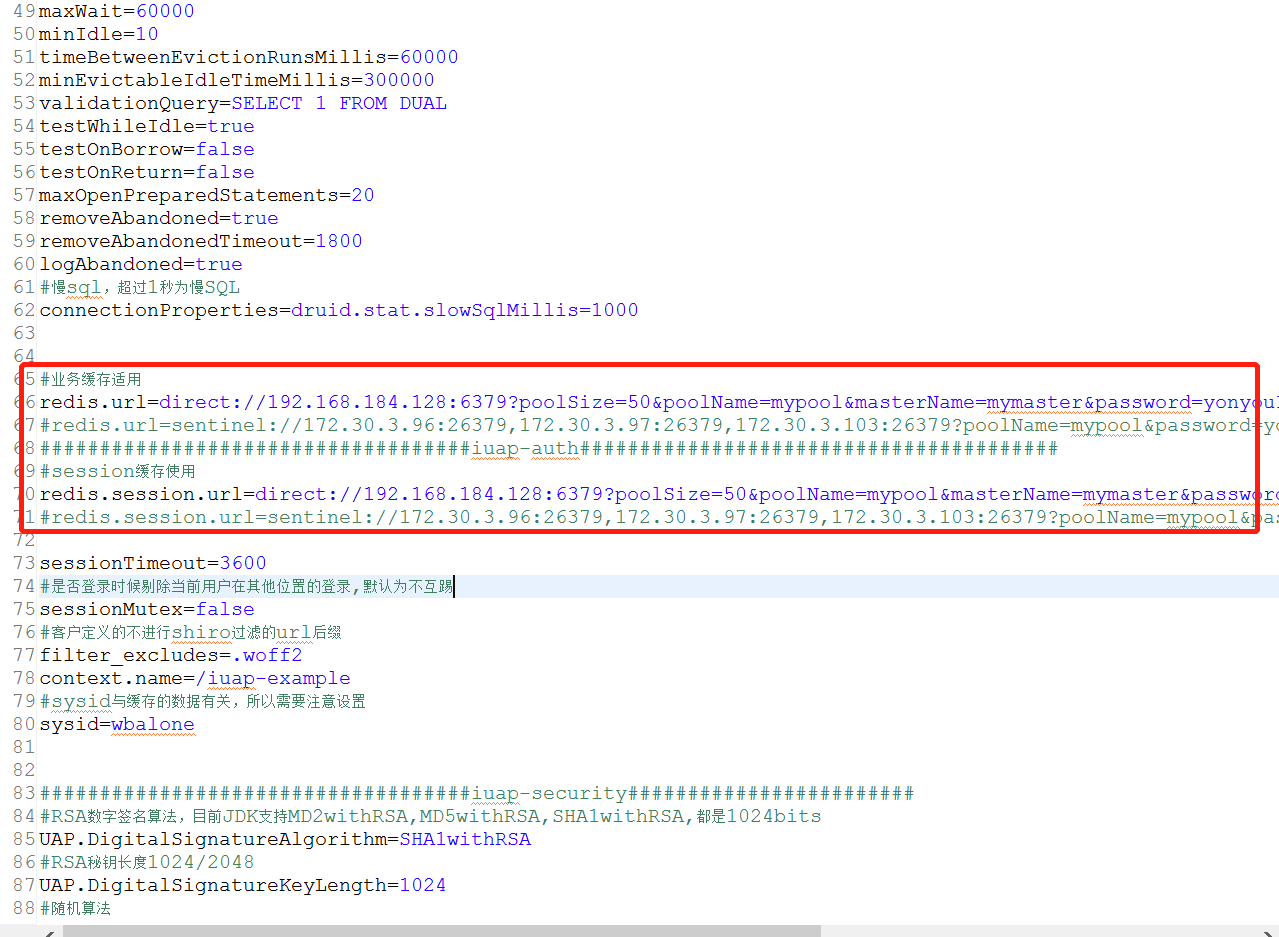
1. jdbc.type改为对应数据库



1. 数据类型
2. 数据库连接改为对应数据库



1. 数据库连接
2. 修改缓存的地址



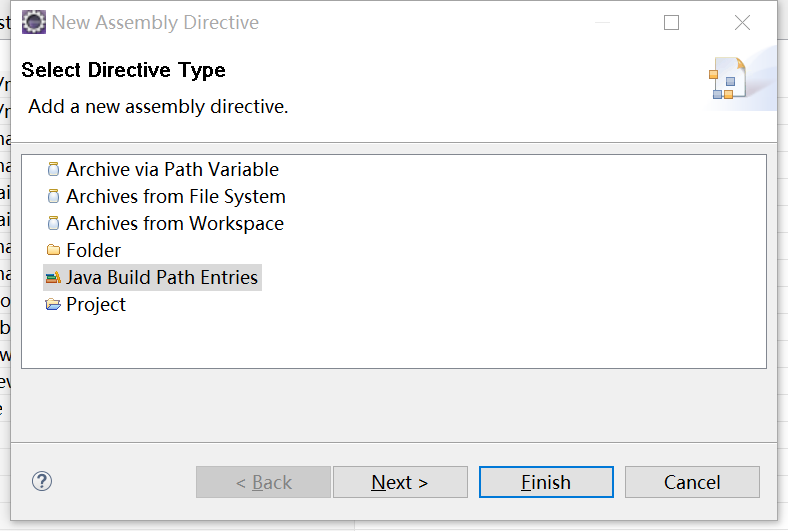
#### applicationContext-persistence.xml

1. sqlSessionFactory改成对应数据库

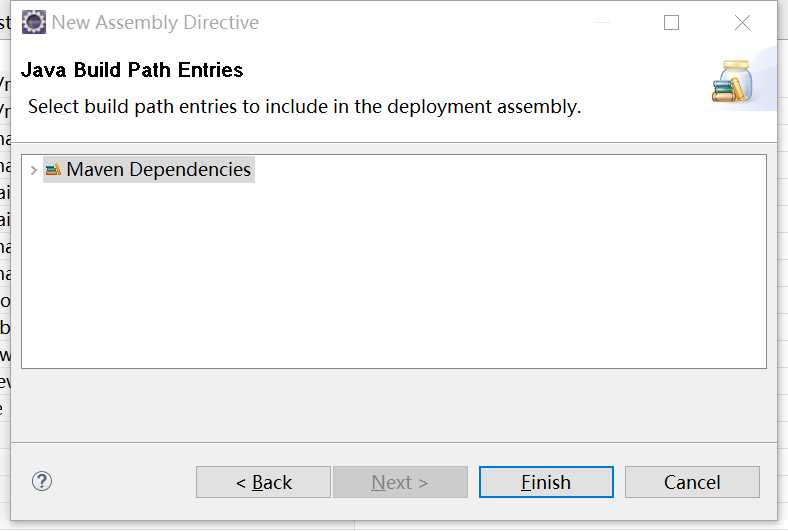


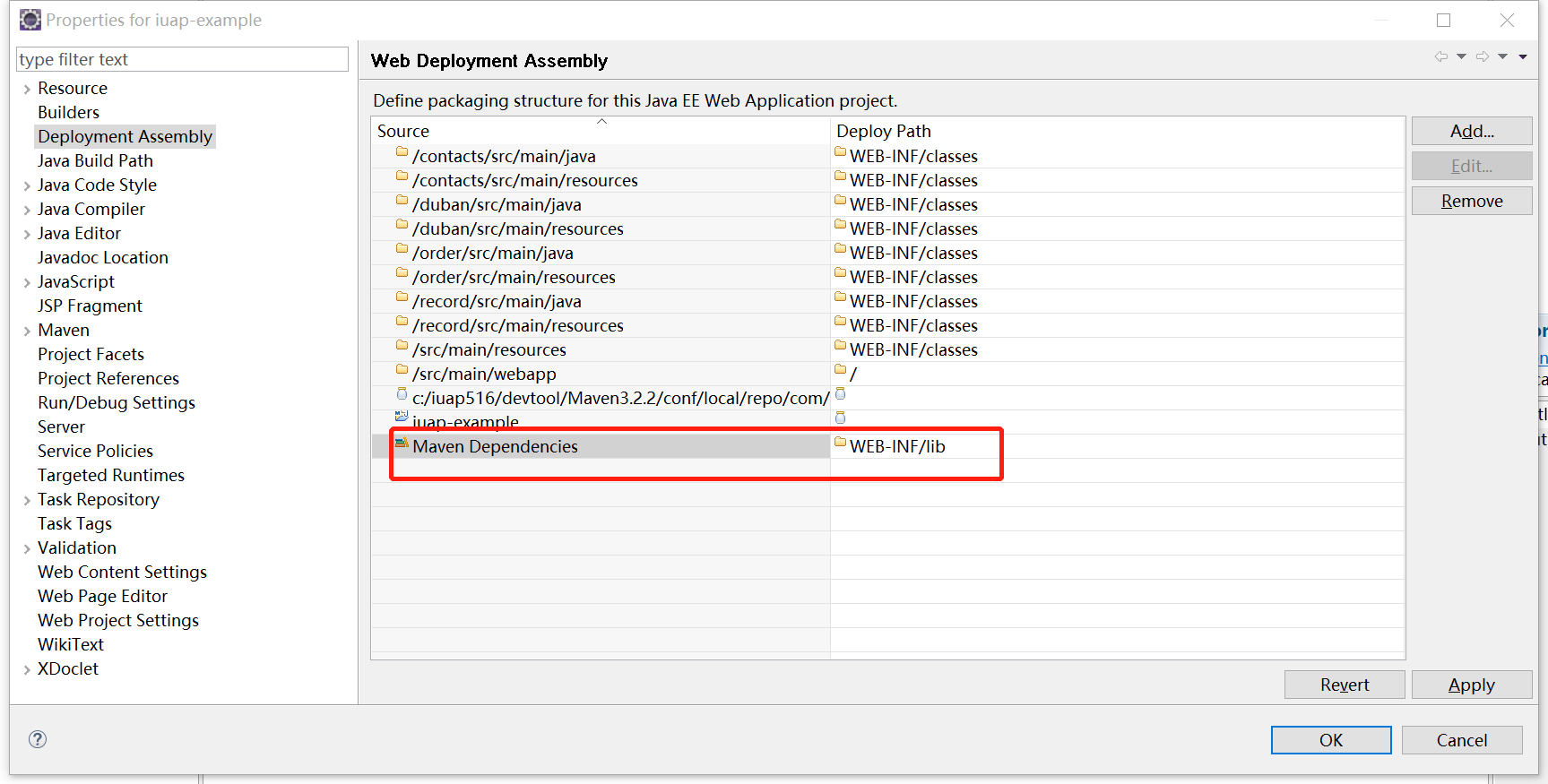
### 设置

注意确认部署设置



1. 添加



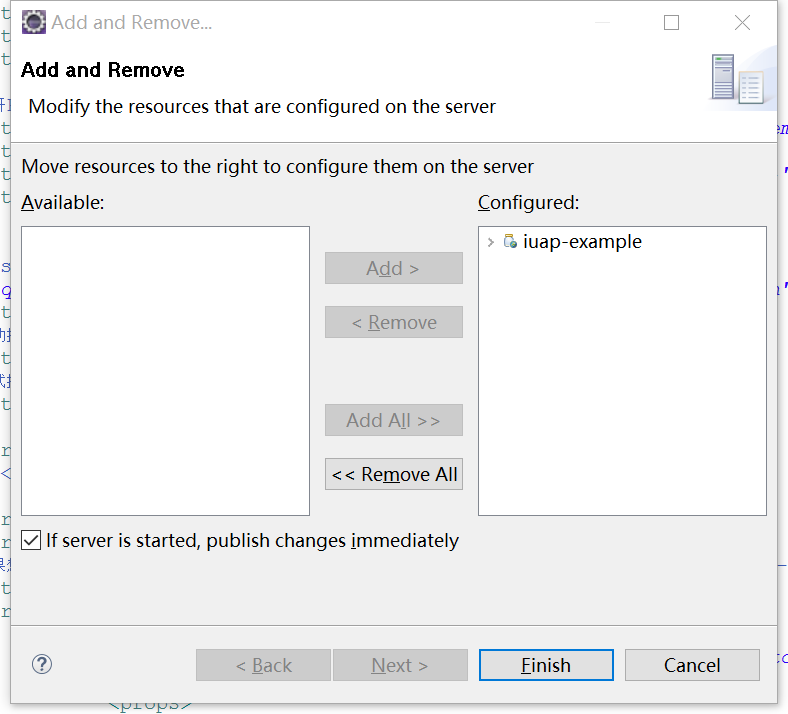


1. 部署设置

## 启动tomcat

### 部署

部署example工程到tomcat中



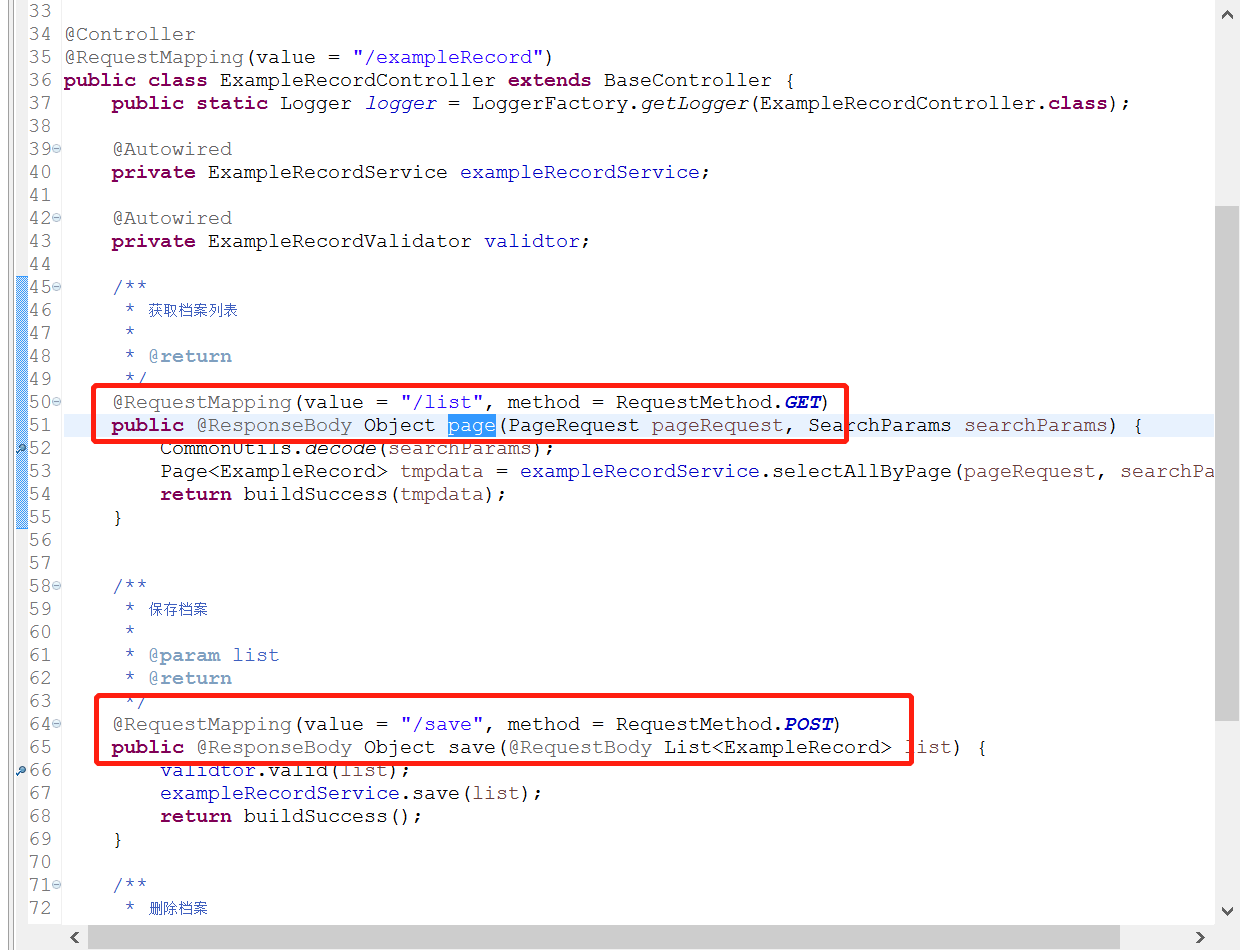
### 启动

采用debug方式启动

## 验证

### 设置

通过ExampleRecordController中设置断点进行验证，在list和save中设置断点。



1. 设置

### 界面验证



1. 验证



1. 验证

### 断点



1. 断点截住则说明已经成功

# 功能节点授权

业务功能开发完毕后需要与IUAP集成，具体包括：（业务节点的注册，业务节点菜单的挂接，业务节点的授权）。当然还有些前置条件需要满足：必须先在系统中创建角色，创建用户，给用户分配角色。

整体步骤如下（注：以下步骤均以管理员身份在“管理中心”操作）

## 创建角色

进入“角色管理”，新增角色。



图 2 新增角色

## 创建用户

进入“用户管理”，新增用户。



图 3 新增用户

## 用户分配角色

进入“角色分配用户”，选中“业务角色1”，分配用户“测试用户1”。



图 4 角色分配用户

## 注册业务节点

进入“功能管理”，点击“新增功能”。



图 5 新增功能

## 业务节点挂接菜单

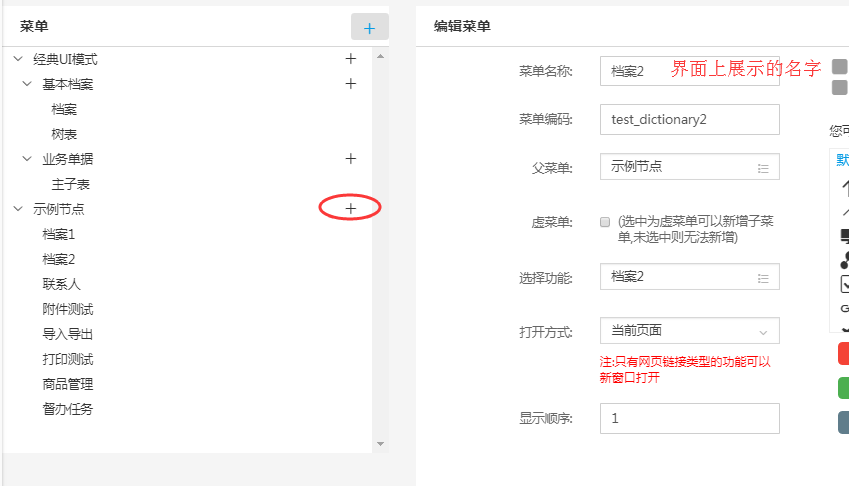


图 6 新增菜单

## 业务节点授权给角色

点击“功能授权”，点击“业务角色1”的功能授权，选中“档案2”这个菜单项，然后点击保存。

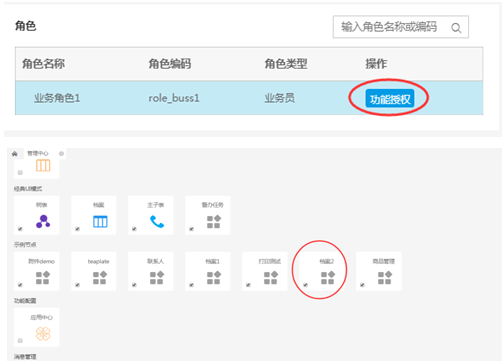
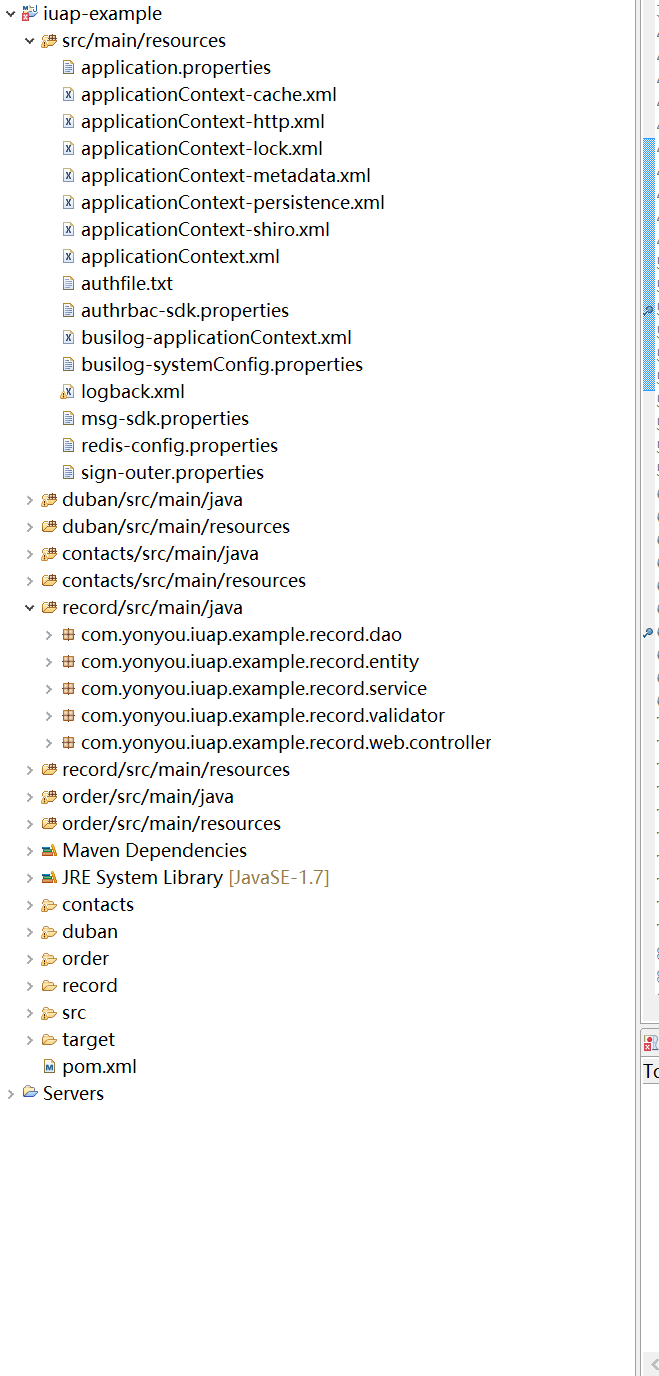


图 7 角色授权功能

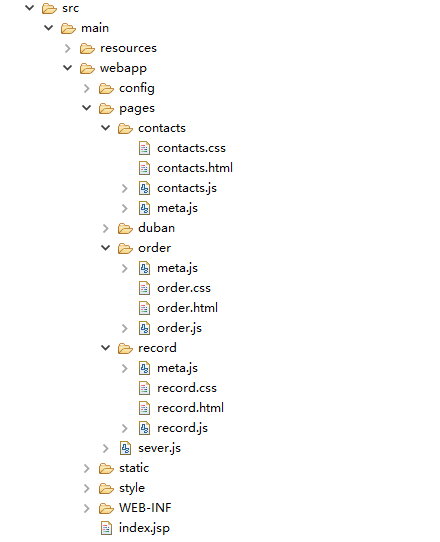
# 代码结构说明

## 示例工程结构





1. 示例工程后台结构



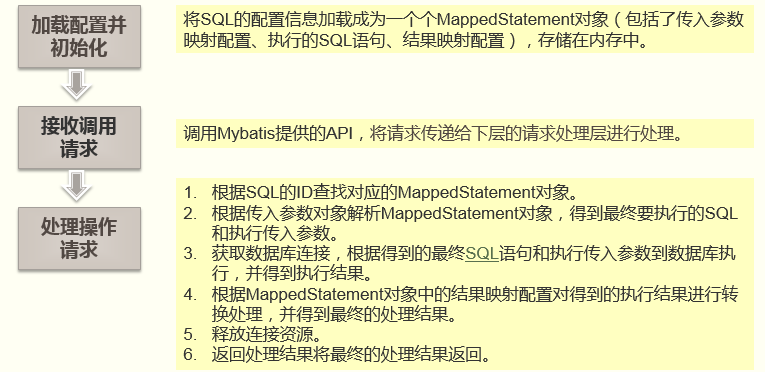
1. 示例工程前台结构

## Mybatis介绍

MyBatis是支持**普通SQL查询，存储过程和高级映射**的优秀持久层框架。

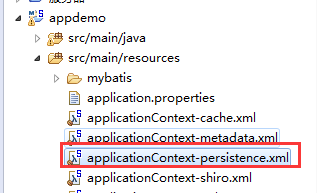
MyBatis消除了几乎所有的JDBC代码和参数的手工设置以及结果集的检索。MyBatis使用简单的XML或注解用于配置和原始映射，将接口和Java的POJOs（Plan Old Java Objects，普通的Java对象）映射成数据库中的记录.

Mybatis运行过程

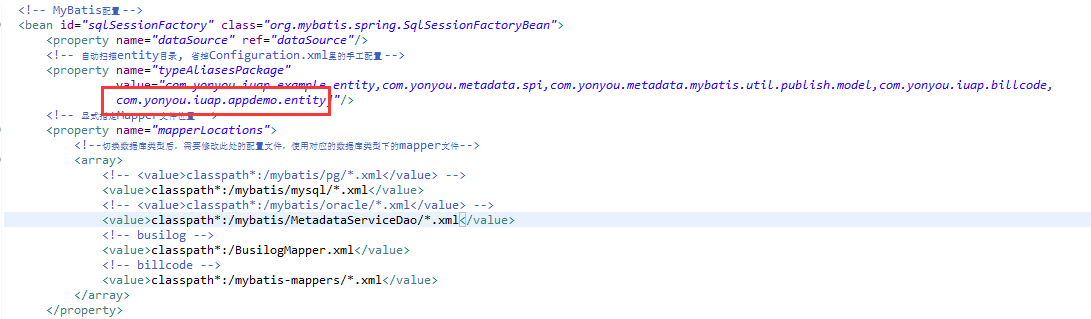


### Mybatis持久化配置

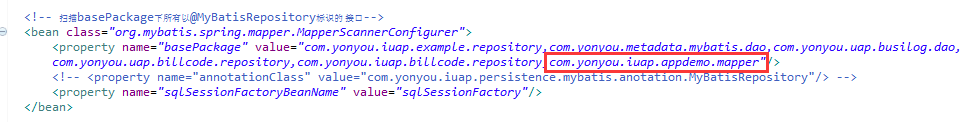
找到持久化配置文件



因为使用了Mybatis的持久化方式，所以需要修改对应的配置信息，增加Mybatis扫描的entity包。同时因为使用了mysql数据库，所以下面配置文件为mysql文件夹下的xml文件



mapper接口类位置，增加我们目前生成的mapper包位置



## 项目配置解析

### Spring集成

iuap 平台集成了Spring框架进行组件的配置和管理，以及Spring MVC作为后端MVC框架，更方便业务开发者进行开发使用。



从pom.xml中可以查看到Spring的版本，集成时此版本务必保持一致。



### SpringMVC

Spring MVC是当前最优秀的MVC框架，自从Spring 2.5版本发布后，由于支持注解配置，易用性有了大幅度的提高。Spring 3.0更加完善，实现了对Struts 2的超越。现在越来越多的开发团队选择了Spring MVC。

#### 配置解析

1.Dispatcherservlet

　　DispatcherServlet是前置控制器，配置在web.xml文件中的。拦截匹配的请求，Servlet拦截匹配规则要自已定义，把拦截下来的请求，依据相应的规则分发到目标Controller来处理，是配置spring MVC的第一步。

　　2.InternalResourceViewResolver

　　视图名称解析器

　　3.以上出现的注解

　　@Controller 负责注册一个bean 到spring 上下文中

　　@RequestMapping 注解为控制器指定可以处理哪些 URL 请求

#### SpringMVC常用注解

　　@Controller

　　负责注册一个bean 到spring 上下文中

　　@RequestMapping

　　注解为控制器指定可以处理哪些 URL 请求

　　@RequestBody

　　该注解用于读取Request请求的body部分数据，使用系统默认配置的HttpMessageConverter进行解析，然后把相应的数据绑定到要返回的对象上 ,再把HttpMessageConverter返回的对象数据绑定到 controller中方法的参数上

　　@ResponseBody

　　该注解用于将Controller的方法返回的对象，通过适当的HttpMessageConverter转换为指定格式后，写入到Response对象的body数据区

　　@ModelAttribute

　　在方法定义上使用 @ModelAttribute 注解：Spring MVC 在调用目标处理方法前，会先逐个调用在方法级上标注了@ModelAttribute 的方法

　　在方法的入参前使用 @ModelAttribute 注解：可以从隐含对象中获取隐含的模型数据中获取对象，再将请求参数 –绑定到对象中，再传入入参将方法入参对象添加到模型中

　　@RequestParam

　　在处理方法入参处使用 @RequestParam 可以把请求参 数传递给请求方法

　　@PathVariable

　　绑定 URL 占位符到入参

　　@ExceptionHandler

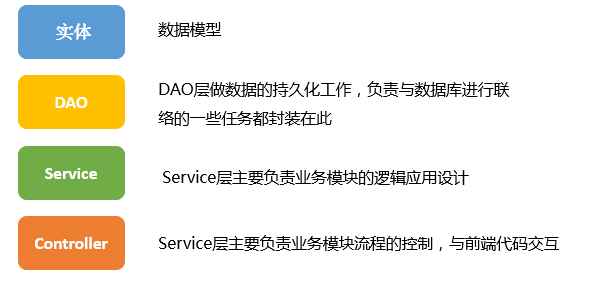
　　注解到方法上，出现异常时会执行该方法

　　@ControllerAdvice

　　使一个Contoller成为全局的异常处理类，类中用@ExceptionHandler方法注解的方法可以处理所有Controller发生的异常

## 后台代码解析

后端代码分为几层：实体、DAO、Service、Controller。



### 配置

iuap平台使用iuap-mybatis作为MyBatis持久化的支持并提供了统一的Spring扫描注解和分页插件来支持分页的实现。

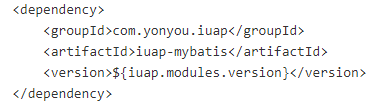
* 功能特性

iuap-mybatis提供了统一的Spring扫描注解和分页插件。通过@MyBatisRepository确定mybatis提供服务的接口，再使用mybatis的xml文件管理实现mybatis服务的sql语句。

分页插件支持MyBatis3.2.0~3.3.0(包含)

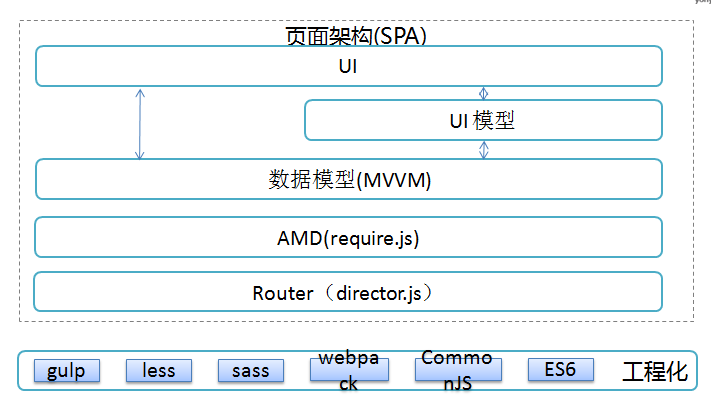
* 配置

在maven中配置



## 前台代码解析

### 前端页面架构



Router——前端路由框架

AMD——模块化管理工具

数据模型——处理企业级复杂数据交互

UI——快速构建前端页面

UI模型——处理UI和数据模型之间的数据通信

### 单页面应用(SPA)

前端界面采用的单页面应用，单页Web应用（single page web application，SPA），就是只有一张Web页面的应用。单页应用程序 (SPA) 是加载单个HTML 页面并在用户与应用程序交互时动态更新该页面的Web应用程序。 浏览器一开始会加载必需的HTML、CSS和JavaScript，所有的操作都在这张页面上完成，都由JavaScript来控制。因此，对单页应用来说模块化的开发和设计显得相当重要。

单页面应用的优点是：

* 速度：更好的用户体验，让用户在web app感受native app的速度和流畅，
* MVC：经典MVC开发模式，前后端各负其责。
* ajax：重前端，业务逻辑全部在本地操作，数据都需要通过AJAX同步、提交。
* 路由：在URL中采用#号来作为当前视图的地址,改变#号后的参数，页面并不会重载

在平台中,实际只有index.html这一个单页面，在这个单页面中，加载了必须的CSS和JavaScript。然后有一个内容区，在点击菜单时，改变地址hash，我们用hash的变化从而推动界面相应的功能变化并在内容区中渲染出来。

那如何根据hash变化调用所需的js方法呢？来让前端路由吧！

### 前端路由

前端路由主要是用在单页面应用中，解决单页面应用里，地址变化，整体页面并不需要重新渲染，只重新渲染局内容。

平台中采用[director.js](https://github.com/flatiron/director)开源框架做为默认的路由框架。基本用法如下：

|  |  |
| --- | --- |
| 1 | var router = Router(); |
| 2 | //定义order路径所对应的操作 | |

|  |  |  |
| --- | --- | --- |
| 3 | router.on('/order', function(){ | |
| 4 | // 具体的实现代码 |

|  |  |  |
| --- | --- | --- |
| 5 | }); | |
| 6 |  |

|  |  |
| --- | --- |
| 7 | router.init(); |

当浏览器地址变为：http://localhost#/ currtype时,会触发上面监听的function，找到对应的币种节点。

在平台前端主页面中，使用了router的相关方法来监听地址变化。同时使用requirejs来做动态加载。从而实现的整个单页面应用的运转。

### AMD模块化规范

AMD是"异步模块定义”。它采用异步方式加载模块，模块的加载不影响它后面语句的运行。所有依赖这个模块的语句，都定义在一个回调函数中，等到加载完成之后，这个回调函数才会运行。

AMD采用require()语句加载模块，但是不同于CommonJS，它要求两个参数：

|  |  |
| --- | --- |
| 1 | require([module], callback); |

第一个参数[module]，是一个数组，里面的成员就是要加载的模块；第二个参数callback，则是加载成功之后的回调函数。例如:

|  |  |  |
| --- | --- | --- |
| 1 | require(['math'], function (math) { | |
| 2 | math.add(2, 3); |

|  |  |
| --- | --- |
| 3 | }); |

math.add()与math模块加载不是同步的，加载完math模块，运用math的add方法，这样浏览器不会发生假死现象。所以很显然，AMD比较适合浏览器环境。

目前，require.js是主要是实现了AMD规范的javascript库。平台中也是使用的require.js做为AMD模块化规范。可参见：http://www.requirejs.cn/ 学习requirejs的用法。

## 代码规范说明

### 后台代码分包

目前示例工程的分包是根据功能模块来区分的，比如record 目录下的代码均为实现单表操作的；contacts目录下的代码为实现树表操作的；order表是实现主子表操作等。

### 包名建议规范

为了简化开发,增加开发速度以及一些不必要的错误，每个功能模块的Java代码包的命名推荐为com.yonyou.iuap 开头；实体类包名建议为entity，maper接口文件包名建议为dao，业务层代码包名为serivce，控制层代码包名为web.controller，数据校验包名为validator；

resource目录下的mybatis配置文件推荐为mybatis/数据库名称/\*.xml 文件格式。

### Java代码建议规范

类名和实体名相关代码建议为驼峰命名方式；

实体类如果为String格式的，在set方法中建议进行trim操作，如下图所示：



为了更少的修改前后台代码，Controller类中查询、保存和删除操作对应的RequestMapping建议为 /list、/save和/del；应保证controller不存在业务代码，业务代码的实现应在service层中实现。

### 前台包名建议规范

前台包建议与后台代码的分包方式一致；前台代码包均放在webapp/pages目录下,根据模块区分;里面包含meta.js，css文件，html文件和js文件。

### 前台代码建议规范

meta.js文件中应该包含后台entity包中所有的实体元素；

html查询条件中的id和name建议为“search\_”+业务实体字段；

# 额外配置

## 修改Nginx配置

|  |
| --- |
| #配置Nginx动静分离(只处理工作台的前端)  location ~ /tenant/.\*\.(html|htm|gif|jpg|jpeg|bmp|png|ico|txt|js|css|woff|ttf)$  {  proxy\_pass http://127.0.0.1:8080;  root html;  index index.html index.htm;  } |

<http://127.0.0.1:8080>为应用平台的访问地址和端口号