项目第一天 环境搭建

十天项目：

第一天：项目背景、需求、环境搭建

第二天：(难度)系统封装

用户管理

第三天：在系统封装基础用户管理开发完成、数据字典、用户认证

第四天：apache poi导入导出(大数据的导入导出)

从第五天开始业务学习

第五天：供货商药品目录模块

第六天：采购单模块

动态分表方法(mybatis操作sql灵活性)

第七天：采购单模块

第八天：采购单模块

第九天：采购单模块  
第十天：统计分析

掌握完整的统计分析解决方案

第一天：项目背景、需求、环境搭建

环境搭建：

Oracle（pl/sql连接oracle）（麻烦）

# 学习方法

一个项目：架构师、高级程序员、一般程序员等参与，项目困难度复杂。

自身定位要准确！

**企业中缺少的：独立模块开发能力的人，根据需求去设计、去编码(dao、service、action、页面)实现，实现功能符合用户需求的！！！**

课堂上要记笔记，

课后遇到问题，把问题总结出来。

# 项目背景

首先了解项目背景，项目的作用？ 解决了什么问题？

医疗行业的问题：药价贵

医疗监管单位

线下交易

医药供货商

医院

**10年前**，由国家卫生部统一指导，以省为单位创建一个平台：

医药招标平台

医药供货商录入药品信息：通用名、剂型、规格、**价格**

**医疗监管单位查询药品信息，从中挑选出中标的药品**

**确定药品中标目录 ：**通用名、剂型、规格、**中标价格（统一价）**

医药供货商

监管单位通过平台进行监管

省级卫生厅

省级医药采购平台

**确定药品中标目录 ：**通用名、剂型、规格、**中标价格（统一价）**

上网供货

上网采购

医院

省级、市级医院

医药供货商

10年后，以省医药采购平台作为指导,作为基础，对乡、镇以下的医药单位（最小医疗单位）上网采购。

市卫生局

市级医药采购平台

把省级平台的药品中标目录引入市级平台内。

**药品中标目录 ：**通用名、剂型、规格、**中标价格（统一价）**

医药供货商

医院

乡、镇以下的医疗单位

（村卫生室）

# 项目概述

药品集中采购系统是利用计算机信息网络技术为医药卫生行业服务的电子交易系统，实现医疗机构上网采购、药品经营企业上网销售、监督单位网上监管，实现药品交易过程的数字化、网络化、透明化。

本系统依托与省级医药招标采购平台，在省级医药集中采购思想的指导下建立的市级医药采购平台，由市卫生监督单位将省级药品目录应用在市平台上，在市卫生局及下属单位监督下，乡、镇以下的卫生室在市平台药品目录范围内进行集中药品采购。

# 用户角色

* 监管单位：

市卫生局(药政科)：负责通过采购平台进行监管，主要监管医院和供货商之间的业务交易情况。

卫生院：乡镇一级医疗单位，由市卫生局直管。负责审核医药下的采购单。

* 医院

卫生室：最小的医疗单位，村卫生室(卫生所)。由由市卫生局直管。

通过平台上网进行药品采购。

* 供货商

药品生产企业、企业销售代理（一级代理，二级代理），称为：药品供货商

接收卫生室药品采购，按时进行药品配送。

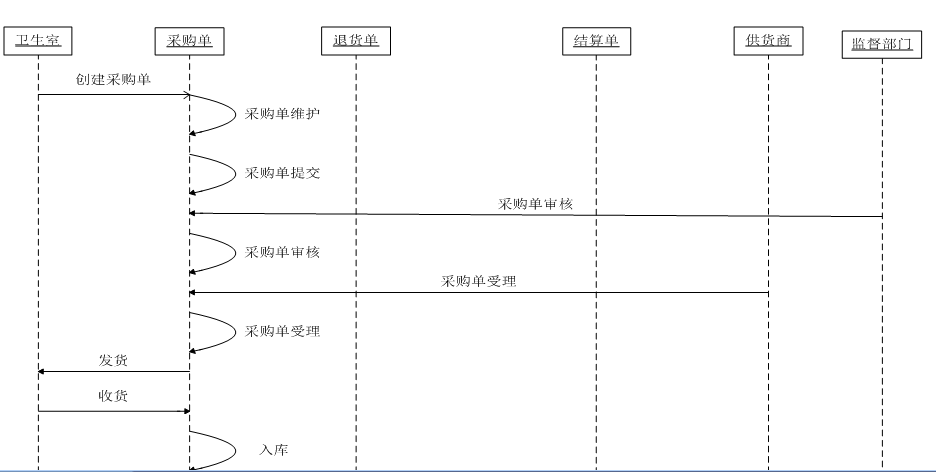
通过平台上网进行药品销售(供应药品)。

* 系统管理员

系统维护、配置、系统备份等。

# 业务流程

不同用户共同完成一个业务流程。



# 业务功能模块

将相似或一致的功能组成一个模块，如下：

采购单管理模块

提交采购单

创建采购单

完整的系统功能如下：



* 药品目录模块：

药品目录维护：由**卫生局**统一管理，记录平台使用药品的中标目录，卫生局在省级目录的基础上进行添加、删除，来组成市级的药品目录。

企业供货目录 ：供货商将自己供应的药品添加企业供货目录，表示供货商要供应这些药品。

* 采购单管理

创建采购单：医院使用

采购单审核：卫生院使用。

* 统计分析

主要给卫生局使用，卫生局通过统计分析功能进行监管。

也给医院、供货商使用，医院、供货商统计自己相关的数据。

* 系统管理

给系统管理员使用。

外部接口：

省级药品目录接口：将省级药品目录 导入到本系统。

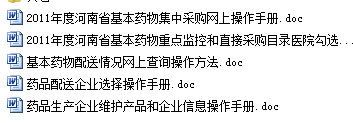
系统管理接口：通过单点登陆系统进行用户认证，将第三方系统中区域管理、系统参数配置等功能接入本系统。

# 需求分析

用户需求的分析：

进行需求调研，一份用户需求调研报告.doc及用户需求文档，用于和用户交流使用。

参考 ：



系统的需求分析：

依据用户需求文档（原始用户需求），进行系统需求分析，成果：“**系统需求规格说明书**”。

系统需求规格说明书：

描述：

需求用例：功能描述、功能接口、前置条件（功能的约束条件）、后置条件（功能影响，主要是数据库操作）。事件流（功能操作流程）

程序员根据需求规格说明书进行开发。

参考：



# 项目配置情况

## 人员配置

|  |  |  |
| --- | --- | --- |
| **角色** | **职责** | **人员** |
| 项目经理 | 项目整体执行控制 | 1 |
| 架构师 | 系统架构搭建  系统集成  技术预研 | 1 |
| 需求分析 | 需求调研、需求分析  编写系统需求规格说明书 | 2(从开发调过来) |
| 开发 | 系统模块开发编码  单元测试  参与系统集成  配合系统测试，修改bug  配合系统产品化工作 | 5 |
| 测试 | 编写测试用例  系统集成测试  系统功能测试  系统性能测试 | 3 |
| 产品化 | 制作系统安装包  制作系统升级包 | 1 |
| 系统维护 | 系统安装调试  用户培训  后期系统维护 | 1 |

## 开发周期

增量模型**：(企业最常用)**

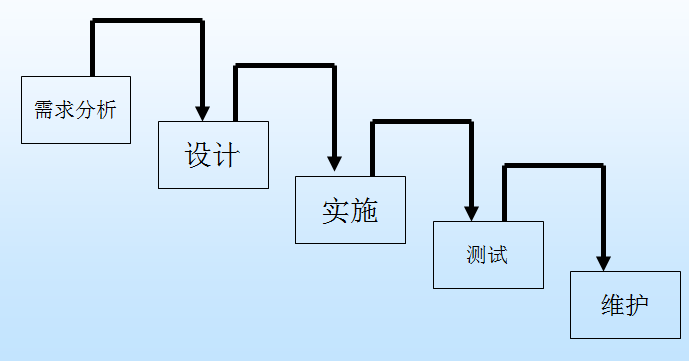
企业中常用的模型的增量模型，当系统很大需要几次开发完成，每次开发给做一个版本，正常进行系统安装升级等工作。

原型模型：

当需求不明确，使用原型模型降低风险。

本系统采用瀑布模型

需求明确，项目周期不长。



2012年2月至2012年8月，项目周期6个月

|  |  |  |  |
| --- | --- | --- | --- |
| 阶段名称 | 开始时间 | 结束时间 | 工期（工作日） |
| 需求阶段 | 2012年2月 | 2012年3月 | 30 |
| 开发阶段 | 2012年3月 | 2012年6月 | 90 |
| 测试阶段 | 2012年5月 | 2012年8月 | **90** |
| 部署上线 | 2012年8月 | 2012年8月 | 30 |

## 系统正式运行的环境。

|  |  |
| --- | --- |
| 网络环境： | 互联网 |
| 硬件平台： | 两台服务器：  一台安装数据库(实际上线用的是oracle集群)  一台安装web服务 |
| 操作系统平台： | Centos 6.5 |
| 数据库平台： | Oracle10G |
| Java环境： | Jdk1.6 |
| Web服务： | Tomcat 7 |

## 开发环境

|  |  |
| --- | --- |
| 网络环境： | 局域网 |
| 硬件平台： | 每人一台台式机或笔记本电脑  两台服务器：(作为系统集成使用，和生产环境一样)  一台安装数据库(oracle)  一台安装web服务 |
| 操作系统平台： | Windows，linux Centos 6.5(集成环境) |
| 数据库平台： | Oracle10G  (企业中是将一台oracle服务器作为开发服务器) |
| Java环境： | Jdk1.6 |
| Web服务： | Tomcat 7 |
| 项目构建： | Maven3.2 |
| 应用软件： | Eclipse4.3(安装maven插件)  Plsqldev(oracle客户端连接工具)  Ssh client(linux远程连接客户端) |

数据库环境

Tns文件：指定连接oracle参数

Oracle客户端oci.dll

Oracle服务器

虚拟机

台式机/笔记本

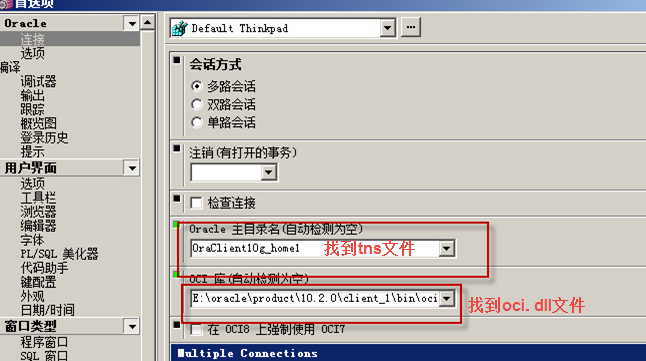
客户端

在客户端进行项目开发

使用pl/sql develper



如果oracle安装在本机，本机已经有Tns文件，oci.dll，只需要在pl/sql上配置一下：



如果本机是64位，pl/sql需要通过32位的oracle客户端连接，在本机安装32位oracle客户端。

安装版本的客户端：



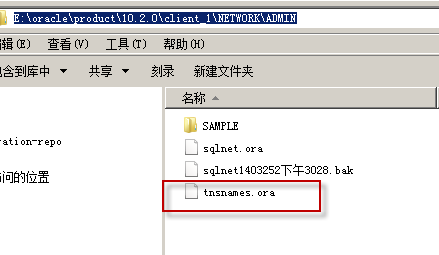
在oracle官方下载免安装的oracle客户端。

Tns文件：

Oracle服务端tns文件位置;



安装oracle客户端，客户端中tns位置：



内容如下：

TEST1 = tns名称

(DESCRIPTION =

(ADDRESS\_LIST =

(ADDRESS = (PROTOCOL = TCP)(HOST = 127.0.0.1)(PORT = 1521))

)

(CONNECT\_DATA =

(SID = yycg)

)

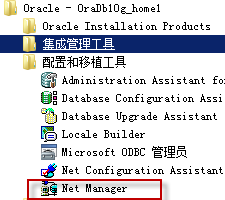
)

127.0.0.1：oracle服务的地址，如果是虚拟机是虚拟机的ip

Sid：Yycg，oracle数据库的标识，一个oralce服务创建多个数据库，每个数据库都有一个SID，默认orcl。

也可以通过图形界面进行配置：

Oracle服务端提供的图形化配置：



如果安装oracle客户端：



建议在安装oracle客户端，选择“管理员”方式。

配置过程详细参考;

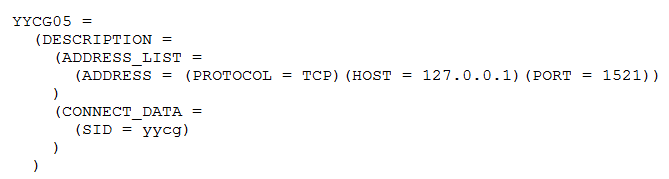


##### 导入数据

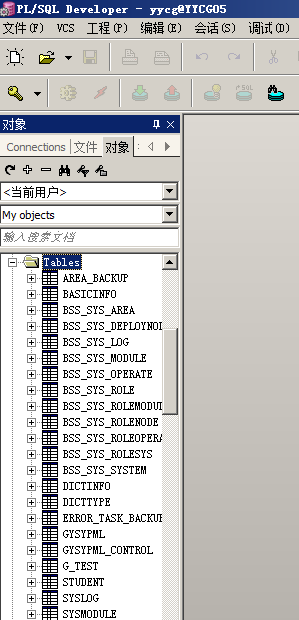
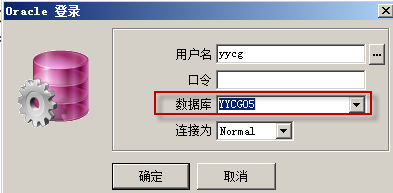
启动oracle：



通过图形化或手动方式配置tns文件，



启动pl/sql：



导入数据具体操作：

参考：开发环境配置.docx

第一步：将oracle\_sql目录 拷贝到d盘



第二步：创建表空间

表空间就是dbf数据文件，存储创建表。

//创建数据表空间

create tablespace yycg0730

logging

datafile 'E:\oracle\product\10.2.0\oradata\yycg\yycg0730.dbf' --表空间数据文件位置在oracle服务器（如果是虚拟机，位置在虚拟机上）

size 32m

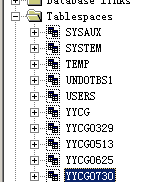
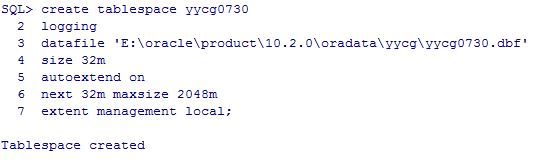
autoextend on

next 32m maxsize 2048m

extent management local;

使用超级管理登陆pl/sql;

选择命令窗口：



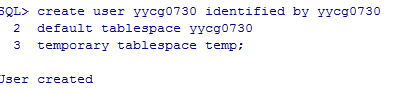
第三步：使用sys创建oracle用户（业务系统连接oracle使用的用户）

创建的用户默认的表空间是yycg0730

create user yycg0730 identified by yycg0730

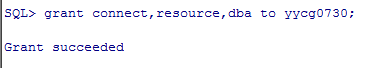
default tablespace yycg0730

temporary tablespace temp;



给用户授权：

grant connect,resource,dba to yycg0730;



第四步：退出sys，使用新创建用户登陆pl/sql

导入数据：

包括：

表结构yycg\_table.sql

初始化数据yycg\_data.sql(在作系统安装包时所需要，系统运行必须有的数据，叫初始数据)

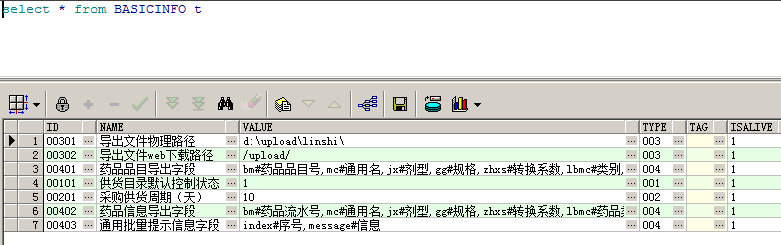
脚本 在oracle\_sql目录

在pl/sql命令窗口，执行如下脚本 ：

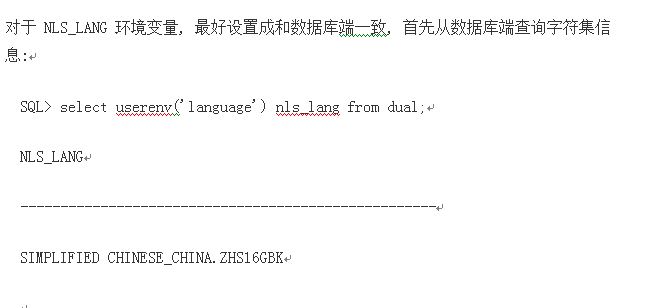
@D:/oracle\_sql/yycg\_table.sql

@D:/oracle\_sql/yycg\_data.sql

查询某个表的数据，是否有乱码：



如果有乱码，需要配置环境变量：



参考：



# 系统设计

包括：

系统概要设计（**必须要做**）

设计内容：主要系统框架、系统内外部接口

内部接口：本平台各各子系统之间的接口，模块之间接口。

外部接口：和其它系统的接口。

网络部署。

成果：“系统概要设计文档”

系统详细设计（不是必须要做）

成果：“系统详细设计文档”

设计内容：类、接口、及方法描述

程序员参考此文档进行编码。

## 系统模块划分

就是系统概要设计中一部分。

这里划分是系统技术架构的模块。依据业务模块进行划分。

以技术架构的纵向拆分方法

依据javaEE分层进行模块划分：

Yycg.dao 持久层

Yycg.service 业务层

Yycg.action 表现层

优点：分层清晰，方便程序员开发。

企业中常见将系统功能进行抽取组成一个子系统，比如系统管理功能，将**系统管理**的功能抽取出来组成一个子系统，让其它系统公用该子系统。

Yycg.dao 持久层

用户管理

系统管理

采购单管理

Yycg.service 业务层

用户管理

系统管理

采购单管理

Yycg.action 表现层

用户管理

系统管理

采购单管理

系统管理（子系统）

缺点：进行系统抽取时需要从各各层拷贝代码，如果有其它功能依赖该抽取功能代码，**系统报错**

模块重用困难，要想在另外一个系统重用模块必需将每个层的代码拷贝出去，模块之间耦合度高。

以业务模块的横向拆分方法

Yycg.usermanager.dao

Yycg.sys. dao

Yycg.cgd. dao

Yycg.usermanager.service

Yycg.sys.service

Yycg.cgd. service

Yycg.usermanager.action

Yycg.sys.action

Yycg.cgd.action

将sys包下的代码全部拷贝出来，组成一个系统管理的子系统

优点：按模块分包更加清晰，方便开发维护，对中大型项目可以采用。

缺点：虽然按模块将包创建，但模块和模块共存与一个工程，每个模块运行都必需将其它模块同时编译、部署，开发效率不高。

**最大的问题：模块之间的依赖没有规范的管理。**

以maven作为架构进行模块划分

Maven是一个优秀的项目管理工具，提高项目管理效率。

使用maven将模块工程化开发，将每个模块创建为一个maven工程。

Maven**工程**（系统管理）

Yycg.sys

Action、service 、dao

Maven**工程**（采购单管理）

Yycg.cgd

Maven**工程**（用户管理）

Yycg.user

Maven提供规范的依赖管理，依赖方将被依赖方的GAV三坐标。（依赖于某个系统的jar包）

将模块的jar发布到仓库之前 ，maven会执行编译、测试 ，通过后将jar发布到仓库，解决了一些编译的错误。

如果将测试做的完善，运行阶段错误基本也可以避免。

优点：每个模块单独创建不同的工程，单个模块运行无需编译其它代码，开发效率提高。模块之间采用maven依赖管理，只要配置好模块之间的依赖，由maven进行管理。

按照maven规范运行整个系统达到一步构建，方便系统部署、测试、运行。

缺点：由于将模块分开多个工程，对于人力不够的小公司，如果采用此模式开发反而会降低开发效率，一个人同时维护多个工程肯定比维护一个工程耗费经历。所以对于小项目不建议采用此模式开发。

## 系统架构

本系统采用maven进行模块划分。

本系统技术架构：springmvc3.1.4+mybatis3.2.3+oracle

技术架构模块

包括：springmvc3.1.4+mybatis3.2.3的jar包

不需要开发。

系统主工程模块

###### Base系统基础模块

包括：系统管理功能，其它业务功能依赖于该base模块

base模块依赖技术架构模块

###### business系统业务模块

business业务模块依赖于base模块。

常用工具类模块

将常用的工具抽取出来组成一个单独的maven工程。

Base模块依赖工具模块。

架构图：

Business业务模块

Utils工具模块

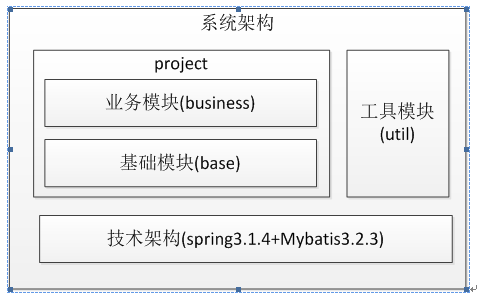
Base系统基础模块

技术架构模块

系统架构及开发规范参考：



为了方便学习，将base和business合并为一个工程。



最终要构建三个maven工程：

系统主工程模块：yycgproject

系统工具类模块：yycgutil

技术架构模块：springmybatis

# Maven环境配置

Maven使用：3.2.1

## 命令行执行

Maven运行，先找用户配置，再找全局配置。

Maven用户配置：

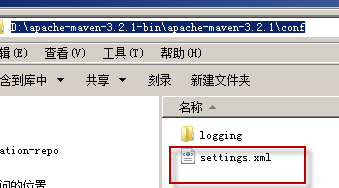
在



有一个setting.xml

Maven全局配置：

在maven安装目录 中



为了方便操作，建议只使用全局配置，不使用.m2下边的setting.xml。

在全局setting.xml中配置本地仓库：



本地仓库：



解压：



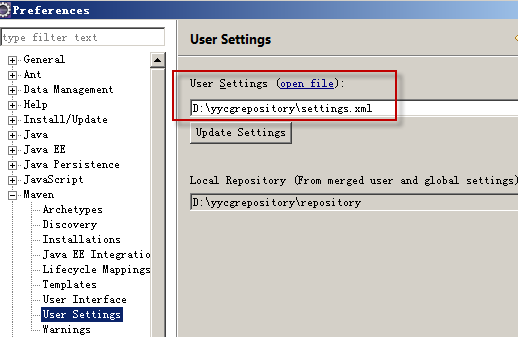
注意：

在命令行执行maven命令，只找全局配置。

## Eclipse执行

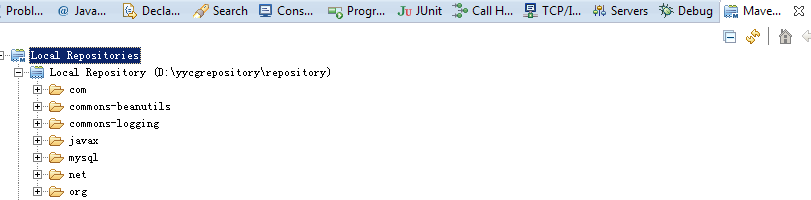
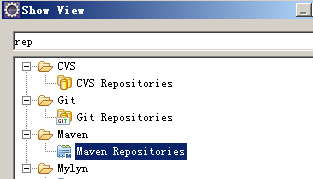
只需要配置maven用户配置：

在本地仓库中创建一个setting.xml，此文件就是eclipse执行maven找maven用户配置文件。



如何判断maven和eclipse整合好？

在eclipse中打开下边的视图：



说明，eclipse和maven整合完成，可以使用本地仓库中的jar包。

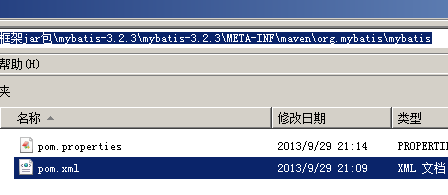
# 系统框架搭建

## 创建技术架构模块

创建一个maven工程，在pom.xml配置springmvc3.1.4和mybatis3.2.3的依赖

在从mybatis官方网站下载mybatis的jar包。

在mybatis目录 中找到pom.xml文件；



得到mybaits所有依赖的jar的GAV。

再找mybaits和spring整合jar包，从mybatis官方网站下载mybaits和spring整合包。



找到pom.xml



在pom.xml中发现mybatis和spring3.1.4整合。

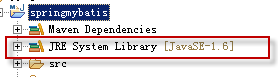
只需要将上边找到的两个pom.xml内容结合放到本系统技术架构模块pom.xml。

确定java编译版本：

在pom.xml文件添加：



右键，工程，



## 工具模块

创建一个maven工程（java工程），yycgutil

先将**工具类**拷贝到src/main/java中。

在pom.xml添加依赖。

## 主工程模块yycgproject

创建一个maven工程（webapp），包括：base(业务基础模块)和business(业务模块)。

## 聚合模块yycgparent

开发在eclipse上开发，将开发后的代码提交svn。

测试从svn上得到代码，编译，打成war包，将war拷贝到测试服务器。

Yycgproject

会打成war

Yycgutil

工具模块生成jar包

聚合工程

将各各工程模块聚合，**maven自动识别模块之间的依赖**。

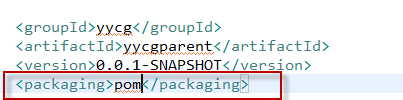
测试人员将各各模块从svn上得下，只需要在聚合工程目录 中，运行maven命令，将各各工程聚合，**生成war包**。

创建聚合工程：yycgparent

负责：将各各工程模块聚合

不需要开发。

创建maven工程，修改pom.xml文件：



加入各各模块：



需要在各各子工程模块，添加父模结点，让子模块继承父模块：

在yycgproject的pom.xml中，添加：



其它模块yycgutil,springmybatis同上。

需要自动生成 war包，需要在yycgproject的pom.xml添加生成war的插件。

<!-- war包生成插件 -->

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-war-plugin</artifactId>

<version>2.4</version>

<configuration>

<warSourceDirectory>src/main/webapp</warSourceDirectory>

</configuration>

</plugin>

为了生成war包之后，自动启动tomcat运行（达到一步构建，执行一个命令：编译、测试、打包、运行。。）

注意：下边要配置正确的tomcat的目录 。

<!-- tomcat运行插件 -->

<plugin>

<groupId>org.codehaus.cargo</groupId>

<artifactId>cargo-maven2-plugin</artifactId>

<version>1.2.3</version>

<configuration>

<container>

<containerId>tomcat7x</containerId>

<home>e:/apache-tomcat-7.0.52</home>

</container>

<configuration>

<type>existing</type>

<home>e:/apache-tomcat-7.0.52</home>

</configuration>

</configuration>

<executions>

<execution>

<id>cargo-run</id>

<phase>install</phase>

<goals>

<goal>run</goal>

</goals>

</execution>

</executions>

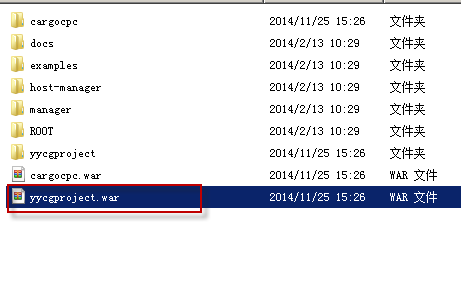
</plugin>

在cmd命令行执行工程聚合，达到一步构建。

cmd进入yycgparent，



在cmd下运行mvn install。



Cmd运行mvn命令，先找用户setting.xml（建议删除了），再去找maven安装目录 中的conf下的setting.xml（全局配置，一定要配置本地仓库）。

运行mvn install之后 ，检查本地仓库，是否三个工程jar发布到本地仓库。

# 主工程模块yycgproject三层构建

基础架构：Springmvc+mybaits

## Base：业务基础模块

主要就是系统管理功能

配置文件

Log4j.properties—》日志配置文件

Db.properties----》连接数据库

配置连接数据库的参数

Mybatis/SqlMapConfig.xml---》 mybatis的核心配置文件

Mybatis配置项

Spring/applicationContext.xml -🡪spring的核心配置文件

配置公用的内容：数据源、事务管理

Spring/applicationContext- base-dao.xml 🡪配置dao

配置SqlSessionFactory，dao（mapper）

Spring/applicationContext-base-service.xml -🡪配置service

配置业务接口

Spring/Springmvc.xml---》配置springmvc

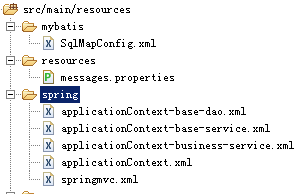
处理器映射器

处理器适配器

视图解析器

拦截器

….

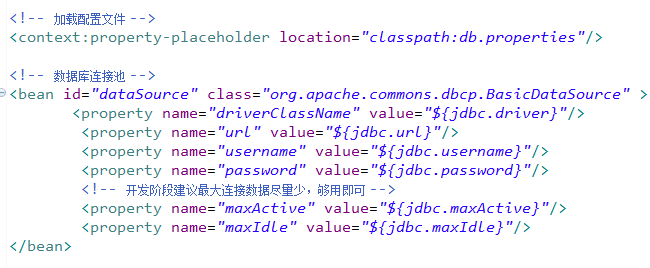


整合Dao：

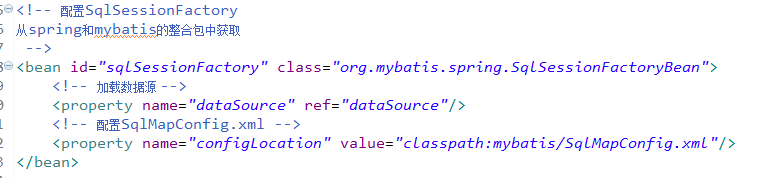
达到目标：将spring和mybatis整合，将spring管理dao接口。

让spring对SqlSessionFactory管理，再由SqlSessionFactory自动生成sqlSession，SqlSession就是dao接口中要用的。

配置：applicationContext.xml



配置applicationContext-dao.xml



##### Spring和mybatis整合,dao开发方法1：

程序员创建dao接口及dao接口的实现类，

Dao接口实现类继承SqlSessionDaoSupport

在spring容器中配置dao接口，并将SqlSessionFactory注入到dao接口实现中，

在dao接口方法中调用this.getSqlSession()获取SqlSession。

##### Mapper动态代理方法：

程序员只需要编写mapper接口(相当于dao接口)，不需要编写 mapper接口的实现类，mybatis提供根据mapper接口和mapper.xml（映射文件）生成mapper接口动态代理对象(mapper接口的实现)。

**具备什么规则生成mapper代理对象：**

Mapper.xml中的namespace等于mapper接口的地址。

Mapper.xml中定义的sql的id（mapped statement的id）等于mapper.java中方法名

Mapper.xml中定义的statement的parametertype等于mapper.java中方法的形参类型。

Mapper.xml中定义的statement的resultType等于mapper.java中方法的返回值类型。

开发两个文件：mapper.java和mapper.xml

* **方式1：使用MapperFactoryBean**

编写mapper.xml：

<!-- SysuserMapperCustom就是mapper接口 -->

<mapper namespace=*"yycg.base.dao.mapper.SysuserMapperCustom"*>

<!-- 根据id查询用户信息 -->

<select id=*"findSysuserById"* parameterType=*"string"*

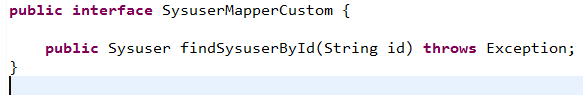
resultType=*"yycg.base.pojo.po.Sysuser"*>

select \* from sysuser where id = #{id}

</select>

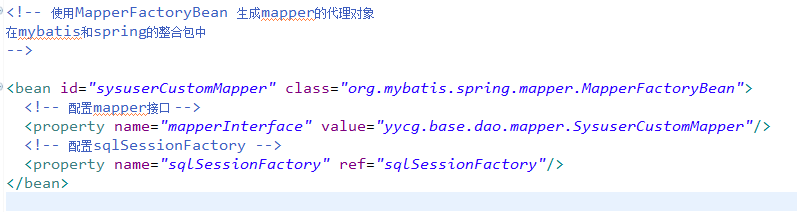
</mapper>

编写mapper.java，



在spring容器中进行配置mapper：

使用**MapperFactoryBean，根据mapper接口生成代理对象。**



注意：mapper.xml和mapper.java同名且在一个目录 ，不需要在SqlMapConfig.xml中加载mapper文件。

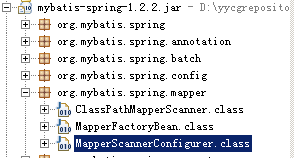
测试方法：

获得spring容器，从容器中得sysuserCustomMapper

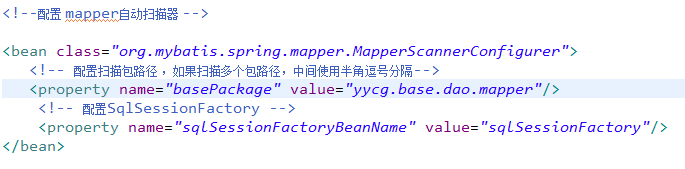
使用**MapperFactoryBean需要在spring容器对每个mapper进行配置，麻烦。**

* 方式2：使用mapper自动扫描器

使用mybaits和spring整合包中提供的mapper扫描器，自动扫描mapper，生成动态代理对象，在spring容器注册。



配置mapper自动扫描器



**开发两个文件：mapper.java和mapper.xml，注意：mapper.xml和mapper.java同名且在一个目录**

测试：

获得spring容器，从容器中得SysuserCustomMapper

本系统采用扫描器方法创建mapper对象。

Mybatis逆向工程

Mybaits官方提供逆向工程，根据数据库表结构生成mapper.xml和mapper.java及po类。

生成的mapper具备了单表的增、删、改、查的功能。

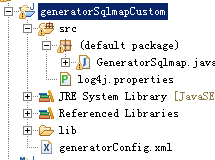


使用方式：



使用java程序+xml配置执行逆向工程。

参考工程：

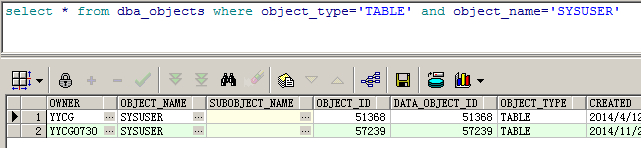


在 generatorConfig.xml配置生成规则

执行GeneratorSqlmap.java

关于mapper.xml中生成内容重复问题：

通过select \* from dba\_objects查询对象的schema



由于该sysuser表有两个schema，生成mapper.xml中内容就重复了。

解决方法：

如果在生成配置文件中指定schema，



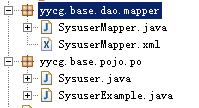
生成mapper.xml，在表名的前边将用户的id硬编码。系统不利于维护的。

**建议：如果在不影响其它用户的前提下，在数据库只保留该表的一个schema。**

**Mapper.xml重新生成时，建议删除后再生成，否则在mapper.xml中追加内容，导致mapper.xml中内容重复。**

逆向工程mapper的使用

在逆向工程中生成mapper文件，将mapper文件拷贝到业务工程中。



将\*.xml拷贝到src/main/resources下。

将\*.java拷贝到src/main/java下。

建议不要在自动生成的文件中添加内容。

调试错误：

org.springframework.beans.factory.BeanCreationException: Error creating bean with name 'sysuserMapper' defined in file [E:\yycg-maven0730\yycgproject\target\classes\yycg\base\dao\mapper\SysuserMapper.class]: Invocation of init method failed; nested exception is java.lang.IllegalArgumentException: org.apache.ibatis.builder.BuilderException: Error parsing Mapper XML. Cause: java.lang.IllegalArgumentException: Result Maps collection already contains value for yycg.base.dao.mapper.SysuserMapper.BaseResultMap

BaseResultMap:在mapper.xml内容重复

基本的操作方法：

//根据主键删除

**public** **void** testDeleteByPrimaryKey() {

sysuserMapper.deleteByPrimaryKey("test009");

}

//插入用户信息

**public** **void** testInsert() {

Sysuser sysuser = **new** Sysuser();

sysuser.setId(UUIDBuild.*getUUID*());//生成uuid主键

sysuser.setUserid("");

sysuser.setUsername("");

sysuserMapper.insert(sysuser);

}

//自定义查询条件查询

**public** **void** testSelectByExample() {

SysuserExample sysuserExample =**new** SysuserExample();

SysuserExample.Criteria criteria = sysuserExample.createCriteria();

//自定义查询条件

criteria.andUsernameEqualTo("test009");

criteria.andGroupidEqualTo("4");

List<Sysuser> list = sysuserMapper.selectByExample(sysuserExample);

System.*out*.println(list.get(0));

}

//根据主键查询用户信息

**public** **void** testSelectByPrimaryKey() {

Sysuser sysuser = sysuserMapper.selectByPrimaryKey("18e6c4bc2f3244bfa146ec6e3f73919a");

System.*out*.println(sysuser);

}

//根据主键更新，传入的po类的属性不为空才更新

**public** **void** testUpdateByPrimaryKeySelective() {

//定义一个新的对象

Sysuser sysuser = **new** Sysuser();

//对象设置id

sysuser.setId("18e6c4bc2f3244bfa146ec6e3f73919a");

//再设置要更新的值

sysuser.setUsername("test009999");

sysuserMapper.updateByPrimaryKeySelective(sysuser);

}

//根据主键更新，不管传入的po类的属性是否为空，都更新

**public** **void** testUpdateByPrimaryKey() {

//先查询对象

Sysuser sysuser = sysuserMapper.selectByPrimaryKey("18e6c4bc2f3244bfa146ec6e3f73919a");

//向对象中设置要更新的值

sysuser.setUsername("test009999");

//执行更新

sysuserMapper.updateByPrimaryKey(sysuser);

}

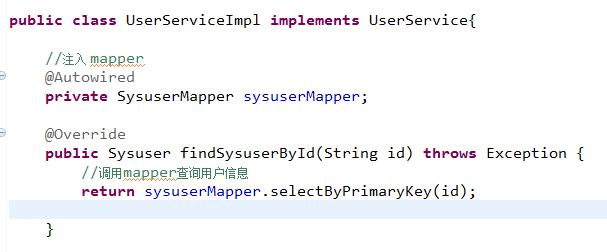
整合service

整合目标：

让spring统一管理service接口，在service中调用mapper。

在service层实现事务控制，使用声明式事务配置方法

编写service :



在applicationContext-service.xml配置：



整合action

整合目标：

在action中调用service接口，从service中获取数据传到页面。

实现系统首页（使用jquery Easyui）。

配置：

Springmvc.xml

<!-- 组件扫描 扫描所有标记@Controller类，由于使用自动扫描所以action类不用在spring配置文件中配置 -->

<context:component-scan base-package=*"yycg.\*\*.action"* />

<!-- 处理器映射器和适配器，可以使用mvc注解驱动 -->

<mvc:annotation-driven/>

<!-- 视图解析器 -->

<bean

class=*"org.springframework.web.servlet.view.InternalResourceViewResolver"*>

<!-- 将jstl的jar包加入工程，默认支持jstl -->

<!-- 前缀和后缀可以去掉的，为了方便开发才加的 -->

<property name=*"prefix"* value=*"/WEB-INF/jsp"* />

<property name=*"suffix"* value=*".jsp"* />

</bean>

Web.xml

加载spring容器

采用统配符方法加载。



Springmvc的前端控制器

<servlet>

<servlet-name>springmvc</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<!-- contextConfigLocation不是必须的， 如果不配置contextConfigLocation， springmvc的配置文件默认在：WEB-INF/servlet的name+"-servlet.xml" -->

<init-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:spring/springmvc.xml</param-value>

</init-param>.

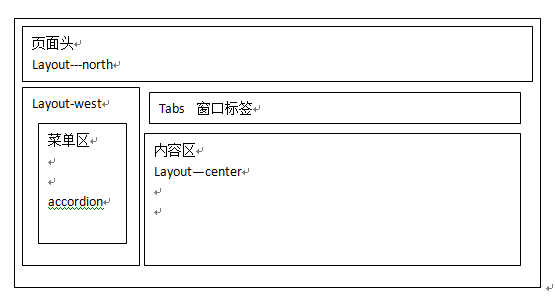
</servlet>

将图片、css、js等表现层目录拷贝到工程。

定义基础的css和基础的js



首页



菜单：配置在menu.json

## Business：业务模块

包括药品目录 、采购单管理、统计分析 。。