**数据库SQL脚本生成器开发**

**手册说明书**

|  |  |  |
| --- | --- | --- |
| **作者** | **联系方式** | **版本** |
| jilongliang | jilongliang@sina.com | V1.0 |
|  |  |  |
|  |  |  |

[一、 简介 3](#_Toc2050)

[1.1前言 3](#_Toc16757)

[1.2作用 3](#_Toc27832)

[二、 实现方式与原理 4](#_Toc21573)

[2.1实现方式 4](#_Toc20751)

[2.2原理 4](#_Toc12815)

[三、 体系与图解 5](#_Toc29669)

[3.1创建表与插数据脚本 6](#_Toc32531)

[3.2添加或修改表脚本 6](#_Toc11759)

[3.3删除表脚本 7](#_Toc10456)

[3.4删除与创建索引脚本 7](#_Toc30614)

[3.5删除与创建序列脚本 7](#_Toc2845)

[3.6删除与创建存储过程脚本 8](#_Toc22192)

[3.7删除与创建视图脚本 8](#_Toc4400)

[四、 开发环境与工具 9](#_Toc15098)

[4.1Nexu中央仓库 10](#_Toc24303)

[4.2 Eclipse 10](#_Toc18310)

[4.3 Tomcat 10](#_Toc9999)

[4.4 JDK 11](#_Toc14458)

[4.5Maven 11](#_Toc16473)

[4.6ER-Ermaster 11](#_Toc20427)

[五、 使用技术体系实现 11](#_Toc23030)

[5.1前端UI 11](#_Toc18657)

[5.1.1 BootStrap 11](#_Toc28426)

[5.1.2 Font Awesome 12](#_Toc4676)

[5.1.3 jQuery 12](#_Toc23590)

[5.1.4 jQuery高亮插件 12](#_Toc17856)

[5.2.5 LayerUI与jBox弹窗技术 13](#_Toc19808)

[5.2.6 colResizable拖拽插件 14](#_Toc23206)

[5.2.7 colResizable拖拽插件 14](#_Toc12817)

[5.2框架与技术 14](#_Toc11025)

[5.2.1 Servlet3.x 14](#_Toc2943)

[5.2.2 SpringMVC 14](#_Toc27736)

[5.2.3 ORM 15](#_Toc22030)

[5.2.4 POI与JXL 15](#_Toc31770)

[5.2.5 JSON 16](#_Toc2209)

[六、 代码实现与效果 16](#_Toc11899)

[6.1Servlet3.x+jdbc实现 16](#_Toc17488)

[6.1.1Pojo 16](#_Toc28564)

[6.1.2Service 16](#_Toc24237)

[6.1.3Util 17](#_Toc22824)

[6.1.4Controler 18](#_Toc793)

[6.1.5其他说明 18](#_Toc9997)

[6.1.6效果 19](#_Toc19466)

[6.2SpringMVC+JdbcTemplate实现 19](#_Toc2040)

[6.2.1Pojo 21](#_Toc31531)

[6.2.2Service 24](#_Toc30091)

[6.2.3DAO 33](#_Toc28597)

[6.2.4Controler 34](#_Toc21962)

[6.2.5前端技术 38](#_Toc13440)

[6.2.6JXL实现 39](#_Toc31090)

[6.2.7Util底层封装代码 41](#_Toc2961)

[6.2.8其他说明 43](#_Toc26048)

[6.2.9功能与效果 49](#_Toc19258)

[七、 个人总结 53](#_Toc3495)

[7.1为什么要总结 54](#_Toc16639)

[7.2为什么记录笔记 55](#_Toc32139)

[7.3其他说明 55](#_Toc12509)

# 简介

## 1.1前言

1**、**由于最近工作一直用Oracle,故对Oracle数据库研究为对象。

2**、**根据工作业务需求实际情况进行功能研发。为什么要开发呢？因为在数据库升级或者迁移的时候，为了保证不同环境不同数据库数据保持同步，故数据库SQL脚本非常作用。比如：数据库主脚本，副脚本，增量脚本。

**3、 什么是主脚本、副脚本、增量脚本呢？**

3.1**、**主脚本指数据库表或存储过程，视图脚本,序列等脚本。

3.2、副脚本指必须执行主脚本之后才执行的脚本。换句话说在没执行主键脚本的情况下，副脚本执行之后会回滚事务失败。

3.3、增量脚本指在执行主脚本或副脚本之后，根据需求对某个表添加/修改约束(主外键约束，长度约束等)，添加/修改字段/添加数据等情况对数据库结构改变处理的一种行为脚本。

## 1.2作用

1**、** 快速产出自定义规则需要的SQL脚本。

2**、**减少人工编写SQL脚本出错率问题，完全通过程序检测SQL准确性。

3**、**帮助开发人员提高SQL编写效率，减少人工编写SQL开发成本问题。

4**、**帮助开发人员节约时间，同时避免繁琐不必要编写SQL的工作。

# 实现方式与原理

## 2.1实现方式

1**、**实现方式分：**正向与逆向实现**。什么是正向与逆行呢【是否有鸡还是有蛋，先后道理同等】

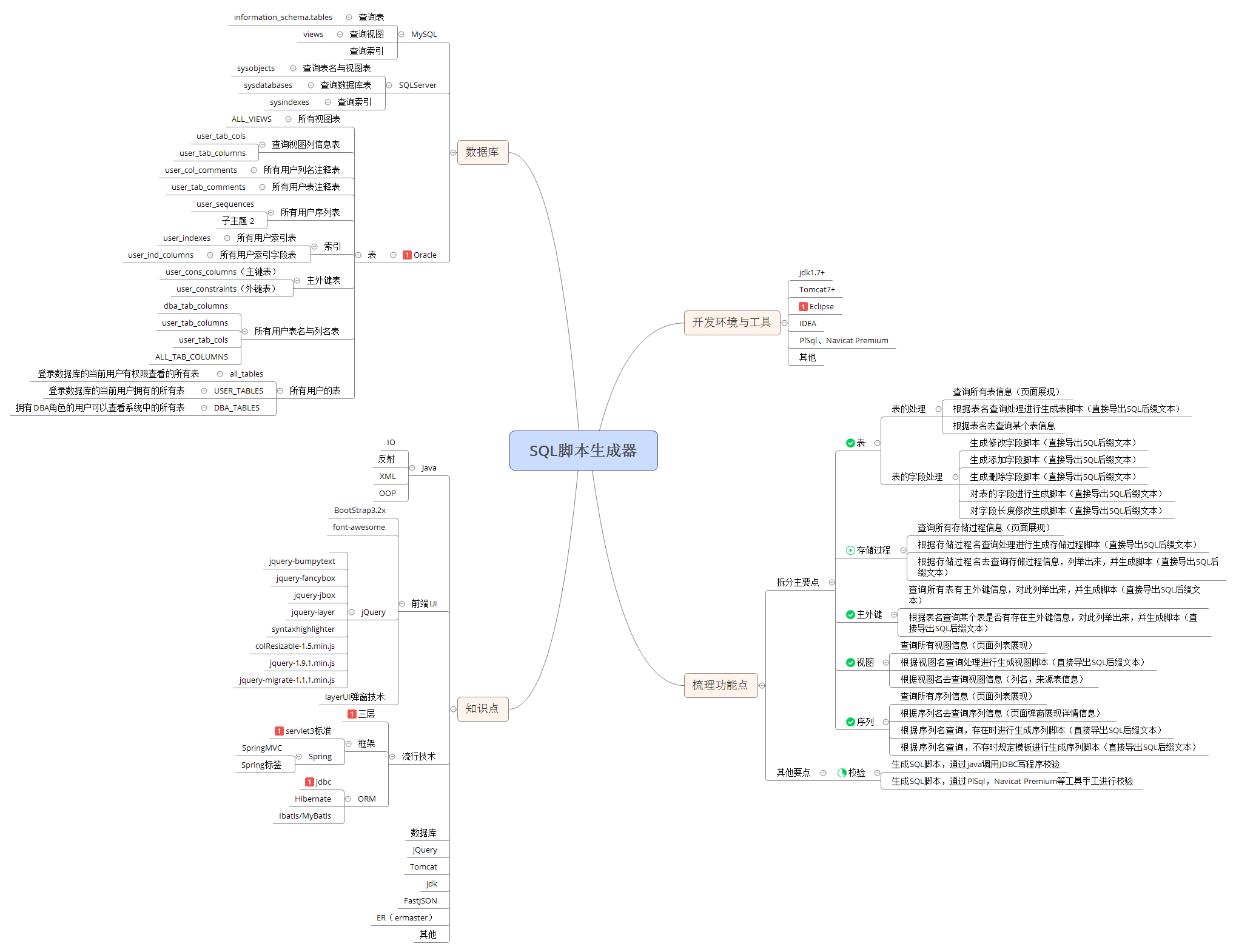
2**、**正向方式：首先把设计好数据库表文档，把所有表的字段属性配置到EXCEL或者CSV格式的文件通过JXL/POI技术去读取文件的字段,再通过其他技术一系列程序处理之后生成所需要的SQL脚本。

3**、**逆向方式：首先有数据库表，然后通过ORM持久化技术连接数据库再读取表的字段等属性出来，再通过其他技术一系列程序处理之后生成所需要的SQL脚本。

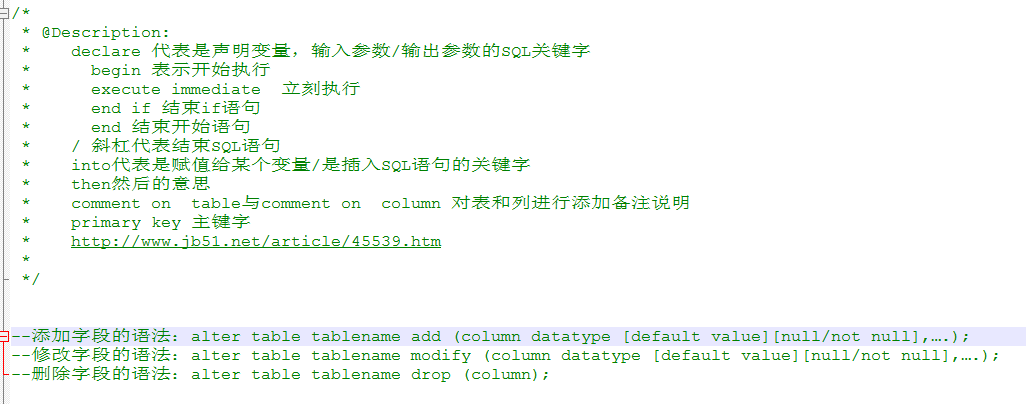
## 2.2原理

对数**据库软件内置核心表或视图**查询出来存储用户行为表结构所有属性信息，对此属性结构信息进行分析与组装所需要SQL脚本。

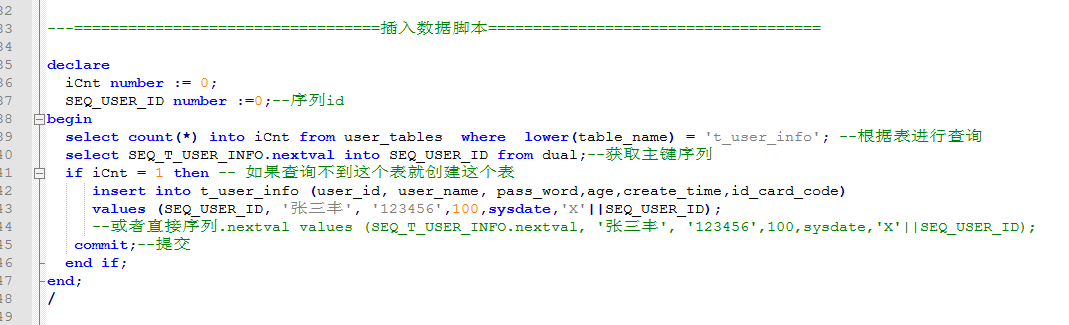
# 体系与图解



## 3.1创建表与插数据脚本



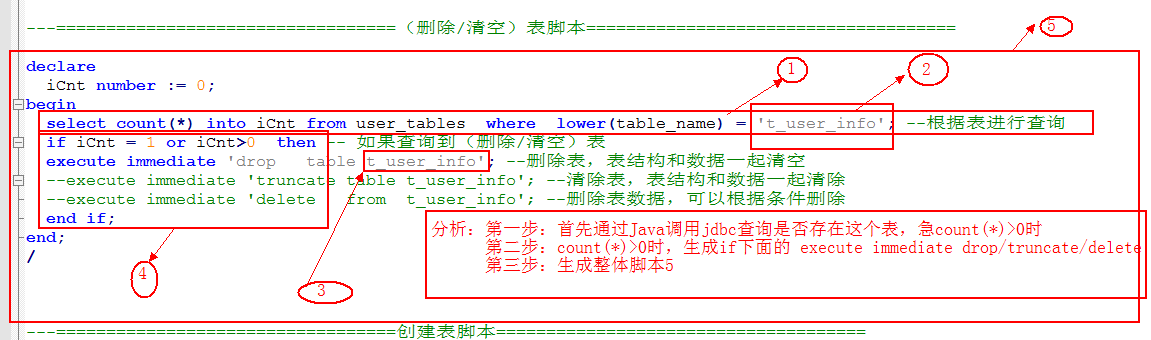




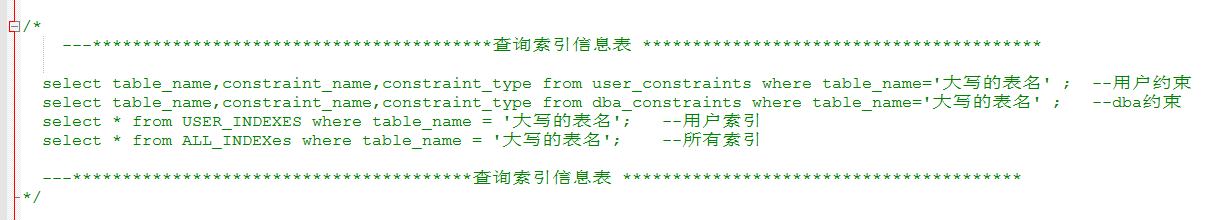
## 3.2添加或修改表脚本

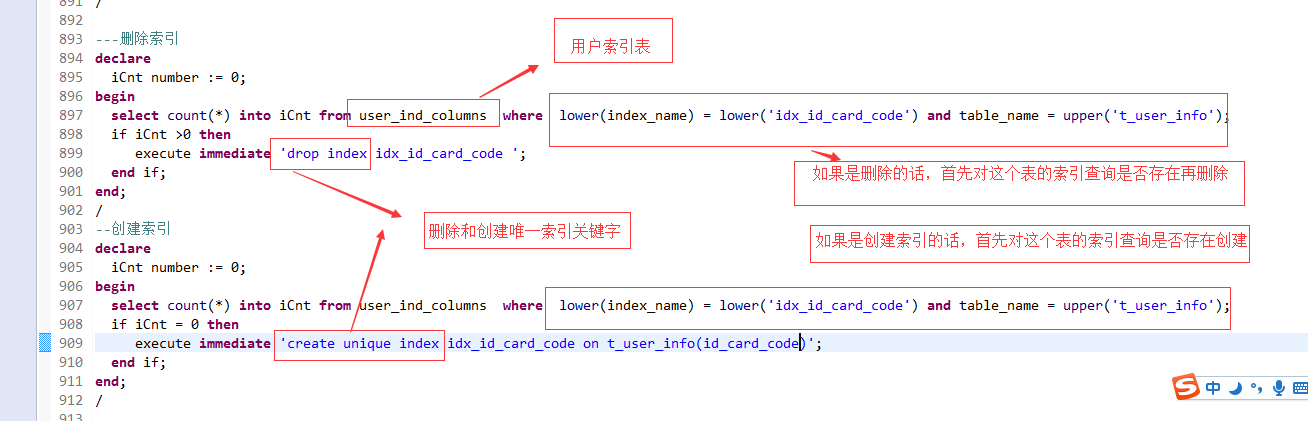


## 3.3删除表脚本



## 3.4删除与创建索引脚本





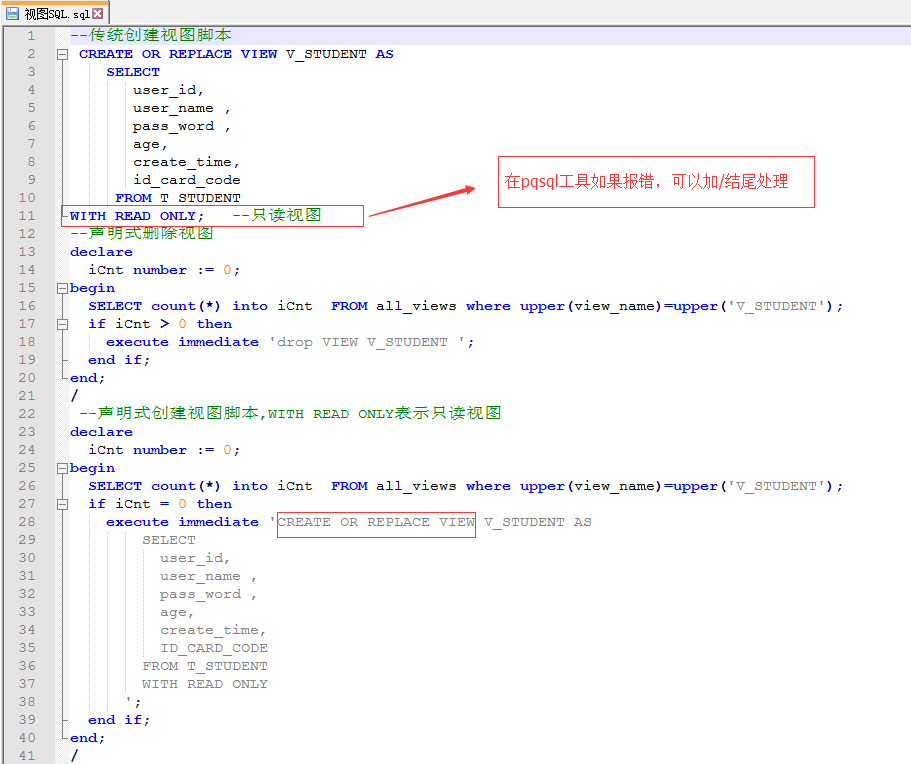
## 3.5删除与创建序列脚本



## 3.6删除与创建存储过程脚本



## 3.7删除与创建视图脚本



# 开发环境与工具

## 4.1Nexu中央仓库

Nexus 是Maven仓库管理器，如果你使用Maven，你可以从Maven中央仓库 下载所需要的构件（artifact），但这通常不是一个好的做法，你应该在本地架设一个Maven仓库服务器，在代理远程仓库的同时维护本地仓库，以节省带宽和时间，Nexus就可以满足这样的需要。此外，他还提供了强大的仓库管理功能，构件搜索功能，它基于REST，友好的UI是一个extjs的REST客户端，它占用较少的内存，基于简单文件系统而非数据库。这些优点使其日趋成为最流行的Maven仓库管理器。

官方网：<http://www.sonatype.org/nexus/>

学习博客：<http://blog.csdn.net/wang379275614/article/details/43940259>

## 4.2 Eclipse

Eclipse 是一个开放源代码的、基于Java的可扩展开发平台。就其本身而言，它只是一个框架和一组服务，用于通过插件组件构建开发环境。幸运的是，Eclipse 附带了一个标准的插件集，包括Java开发工具（Java Development Kit,JDK）。

官方网：<http://www.eclipse.org/downloads/>

## 4.3 Tomcat

Tomcat是Apache 软件基金会（Apache Software Foundation）的Jakarta 项目中的一个核心项目，由Apache、Sun 和其他一些公司及个人共同开发而成。由于有了Sun 的参与和支持，最新的Servlet 和JSP 规范总是能在Tomcat 中得到体现，Tomcat 5支持最新的**Servlet2.4和JSP2.0规范**。因为Tomcat 技术先进、性能稳定，而且免费，因而深受Java 爱好者的喜爱并得到了部分软件开发商的认可，成为目前比较流行的Web 应用服务器。

Tomcat服务器是一个免费的开放源代码的Web 应用服务器，属于轻量级应用服务器，在中小型系统和并发访问用户不是很多的场合下被普遍使用，是开发和调试JSP 程序的首选。对于一个初学者来说，可以这样认为，当在一台机器上配置好Apache 服务器，可利用它响应HTML（标准通用标记语言下的一个应用）页面的访问请求。实际上Tomcat部分是Apache服务器的扩展，但它是独立运行的，所以当你运行tomcat 时，它实际上作为一个与Apache 独立的进程单独运行的。诀窍是，当配置正确时，Apache 为HTML页面服务，而Tomcat 实际上运行JSP页面和Servlet。另外，Tomcat和IIS等Web服务器一样，具有处理HTML页面的功能，另外它还是一个Servlet和JSP容器，独立的Servlet容器是Tomcat的默认模式。不过，Tomcat处理静态HTML的能力不如Apache服务器。目前Tomcat最新版本为9.0

**官方网**：http://tomcat.apache.org/

## 4.4 JDK

JDK是 Java 语言的软件开发工具包，主要用于移动设备、嵌入式设备上的java应用程序。JDK是整个java开发的核心，它包含了JAVA的运行环境，JAVA工具和JAVA基础的类库

官方网：http://www.oracle.com/technetwork/java/javase/downloads/index.html

## 4.5Maven

Apache Maven是一个软件项目管理和理解工具。 基于项目对象模型（POM）的概念，Maven可以从中央信息管理项目的构建，报告和文档。

官方网：<http://maven.apache.org/>

学习博客：

<https://yq.aliyun.com/articles/28591>

<http://www.yiibai.com/maven/maven_project_documents.html>

## 4.6ER-Ermaster

ERMaster是一个用于设计ER模型图的Eclipse插件。提供的功能包括：从数据库导入关系生成ER图，导出设计图，导出DDL数据定义语句等,详细教程请看

官方网：<http://ermaster.sourceforge.net/>

# 使用技术体系实现

## 5.1前端UI

### 5.1.1 BootStrap

BootStrap，来自 Twitter，是目前很受欢迎的前端框架。Bootstrap 是基于 HTML、CSS、JAVASCRIPT 的，它简洁灵活，使得 Web 开发更加快捷。它由Twitter的设计师Mark Otto和Jacob Thornton合作开发，是一个CSS/HTML框架。Bootstrap提供了优雅的HTML和CSS规范，它即是由动态CSS语言Less写成。Bootstrap一经推出后颇受欢迎，一直是GitHub上的热门开源项目，包括NASA的MSNBC（微软全国广播公司）的Breaking News都使用了该项目。国内一些移动开发者较为熟悉的框架,如WeX5前端开源框架等，也是基于Bootstrap源码进行性能优化而来。

官方网: <http://www.bootcss.com/>

BootStrap图标：<http://v3.bootcss.com/components/>

### 5.1.2 Font Awesome

Font Awesome为您提供可缩放的矢量图标，您可以使用CSS所提供的所有特性对它们进行更改，包括：大小、颜色、阴影或者其它任何支持的效果。

官方网：<http://fontawesome.dashgame.com/>

### 5.1.3 jQuery

jQuery是一个快速、简洁的JavaScript框架，是继Prototype之后又一个优秀的JavaScript代码库（或JavaScript框架）jQuery设计的宗旨是“**write Less，Do More**”，即倡导写更少的代码，做更多的事情。它封装JavaScript常用的功能代码，提供一种简便的JavaScript设计模式，优化HTML文档操作、事件处理、动画设计和Ajax交互。jQuery的核心特性可以总结为：具有独特的链式语法和短小清晰的多功能接口；具有高效灵活的css选择器，并且可对CSS选择器进行扩展；拥有便捷的插件扩展机制和丰富的插件。jQuery兼容各种主流浏览器，如IE 6.0+、FF 1.5+、Safari 2.0+、Opera 9.0+等

官方网: <http://jquery.com/>

### 5.1.4 jQuery高亮插件

一款优秀的JavaScript代码高亮插件，将会帮助你渲染任何一种编程语言，包括一些关键字的着色，以及每行代码的缩进等，非常有效地帮你实现在网页上的代码编辑和展示。

在项目中我首选了SyntaxHighlighter

1、SyntaxHighlighter–最优秀的JavaScript代码高亮插件SyntaxHighlighter 是一款完全基于JavaScript的代码高亮插件，SyntaxHighlighter可以对大部分编程语言进行着色渲染，而且代码高亮的性能也非常不错。SyntaxHighlighter 可以自定义主题文件，在初始化的时候指定自己喜欢的主题即可。

官方网站：<http://alexgorbatchev.com/SyntaxHighlighter/>

2、Google Code Prettify–自由地JavaScript代码高亮插件Google Code Prettify是一款由Google推出的JavaScript代码高亮插件，Google Code Prettify可以对C/C++，Java，Python，Ruby，PHP，VisualBasic，AWK，Bash，SQL，HTML，XML，CSS，JavaScript，Makefiles和部分Perl编程语言代码高亮着色。

官方网站：<http://code.google.com/p/google-code-prettify/> 需要翻墙下载

3、Highlight.js –多风格JavaScript代码高亮插件Highlight.js是一个用于在任何web页面上高亮着色显示各种示例源代码语法的JavaScript项目。

官方网站：<https://highlightjs.org/>

4. Prism.JS - 轻量级JavaScript代码高亮插件Prism.JS是目前最为轻量级的JavaScript代码高亮插件，他压缩后只有2KB的大小，Prism.JS也支持大部分流行的编程语言，并且支持多种主题样式，开发者只需要引用CSS文件和JS文件即可完成。

官方网站：<http://prismjs.com/>

1. jQuery.Syntax–jQuery轻量级代码高亮插件

这款代码高亮插件是基于jQuery的，同样也是轻量级的，渲染速度非常快。同时jQuery.Syntax还具有wordpress的插件应用，对于个人博主来说，使用wordpress和jQuery.Syntax可以完美实现0代买的高亮功能。

官方网站：<http://www.codeotaku.com/projects/jquery-syntax/index.en>

6、DlHighlight–jQuery简单高效代码高亮插件

DlHighlight是基于JavaScript的代码高亮插件，非常简单，目前只支持JavaScript, CSS, XML, HTML。

官方网站：<http://mihai.bazon.net/projects/javascript-syntax-highlighting-engine>

7、Rainbow.js – 可扩展的JavaScript代码高亮插件

Rainbow 是JavaScript开发的语法高亮工具。被设计为轻量级（压缩后仅1.4 kb），使用简单，可扩展。语法高亮主题完全通过CSS定义。基于正则表达式实现。

官方网站：<http://craig.is/making/rainbows>

### 5.2.5 LayerUI与jBox弹窗技术

layer是一款web弹层组件，致力于服务各个水平段的开发人员。

官方网: <http://layer.layui.com/>

<http://www.5imvc.com/scripts/jbox/jbox-demo.html>

官方API: <http://layer.layui.com/api.html>

### 5.2.6 colResizable拖拽插件

colResizable是一个jQuery插件，用于调整手动拖动HML的Table标签列。 它兼容鼠标和触摸设备，并具有一些不错的功能，如页面刷新或回发后的布局持久性。这个插件在处理大表提高可用性时非常方便。 它的体积很小（colResizable 1.0只有2kb），它与所有主流浏览器（IE7 +，Firefox，Chrome和Opera）完全兼容。

官方网：<http://www.bacubacu.com/colresizable/>

### 5.2.7 colResizable拖拽插件

bumpyText一款能让文字跳舞的jQuery文字插件，鼠标划过文字即可看到效果。兼容主流浏览器！

Github: <https://github.com/alexanderdickson/bumpyText>

## 5.2框架与技术

### 5.2.1 Servlet3.x

Servlet 3.0 作为 Java EE 6 规范体系中一员，随着 Java EE 6 规范一起发布。该版本在前一版本（Servlet 2.5）的基础上提供了若干新特性用于简化 Web 应用的开发和部署。其中有几项特性的引入让开发者感到非常兴奋，同时也获得了 Java 社区的一片赞誉之声

推荐学习博客

<https://www.ibm.com/developerworks/cn/java/j-lo-servlet30/>

<http://jinnianshilongnian.iteye.com/category/255452>

### 5.2.2 SpringMVC

Spring MVC属于SpringFrameWork的后续产品，已经融合在Spring Web Flow里面。Spring 框架提供了构建 Web 应用程序的全功能 MVC 模块。使用 Spring 可插入的 MVC 架构，从而在使用Spring进行WEB开发时，可以选择使用Spring的SpringMVC框架或集成其他MVC开发框架，如Struts1，Struts2等。而且Spring体系非常庞大，这里不做详细描述，想继续了解和学习请到官方网。

Spring官方网 <http://spring.io>

推荐学习博客：<http://jinnianshilongnian.iteye.com/category/231099>

### 5.2.3 ORM

对象关系映射（英语：(Object Relational Mapping，简称ORM，或O/RM，或O/R mapping），是一种程序技术，用于实现面向对象编程语言里不同类型系统的数据之间的转换。从效果上说，它其实是创建了一个可在编程语言里使用的“虚拟对象数据库”。面向对象是从软件工程基本原则（如耦合、聚合、封装）的基础上发展起来的，而关系数据库则是从数学理论发展而来的，两套理论存在显著的区别。为了解决这个不匹配的现象，对象关系映射技术应运而生。对象关系映射（Object-Relational Mapping）提供了概念性的、易于理解的模型化数据的方法。

ORM方法论基于三个核心原则：

简单：以最基本的形式建模数据。传达性：数据库结构被任何人都能理解的语言文档化。精确性：基于数据模型创建正确标准化的结构。

典型地，建模者通过收集来自那些熟悉应用程序但不熟练的数据建模者的人的信息开发信息模型。建模者必须能够用非技术企业专家可以理解的术语在概念层次上与数据结构进行通讯。建模者也必须能以简单的单元分析信息，对样本数据进行处理。ORM专门被设计为改进这种联系。比如：（JDBC，JdbcTemplate，Hibernate ，Ibatis/MyBatis）等

本项目框架在3年前工作中用到JdbcTemplate在资料的时候看到oschina大牛的一篇文章，故下决心通过学习整理一个SpringMVC工程作为技术积累。

推荐博客：<https://www.oschina.net/code/snippet_1245103_33821>

### 5.2.4 POI与JXL

简述：POI与JXL都是一个处理Excel文档的技术。个人认为：从两者的使用过程中的角度来看，JXL相对POI会轻巧，占空间内存少，Jar只有一个，上手快。不过从知识上的支持没POI友好，实现方式没POI强大。毕竟POI是Apache下的工程项目。建议可以根据自己的情况进行选择与学习。

1、POI官方网: <http://poi.apache.org/>

2、JXL官方网: <http://jxl.sourceforge.net/>

3、入门例子：

<https://github.com/jilongliang/excel-doc-pdf>

<https://github.com/jilongliang/JL_OutExcel>

### 5.2.5 JSON

fastjson gson jackjson json-lib org.json

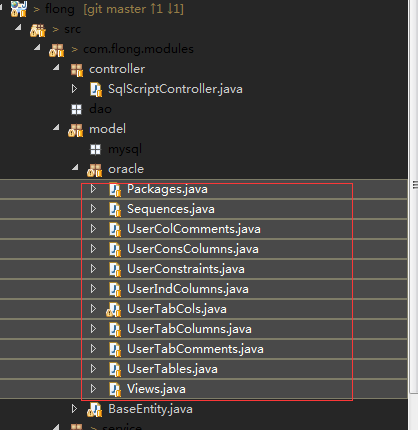
JSON工程例子代码 <https://github.com/jilongliang/json>

博客文章：<http://blog.csdn.net/jilongliang/article/category/2813267>

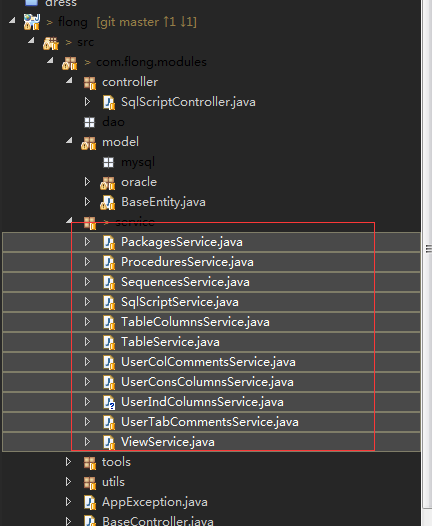
# 代码实现与效果

## 6.1Servlet3.x+jdbc实现

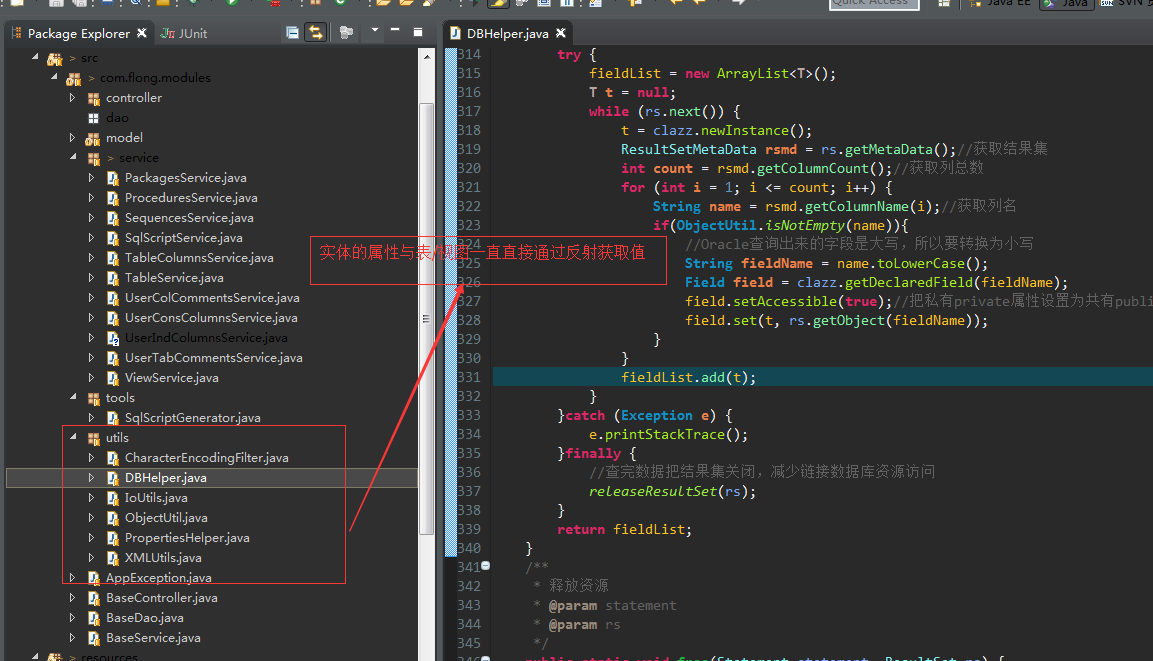
### 6.1.1Pojo



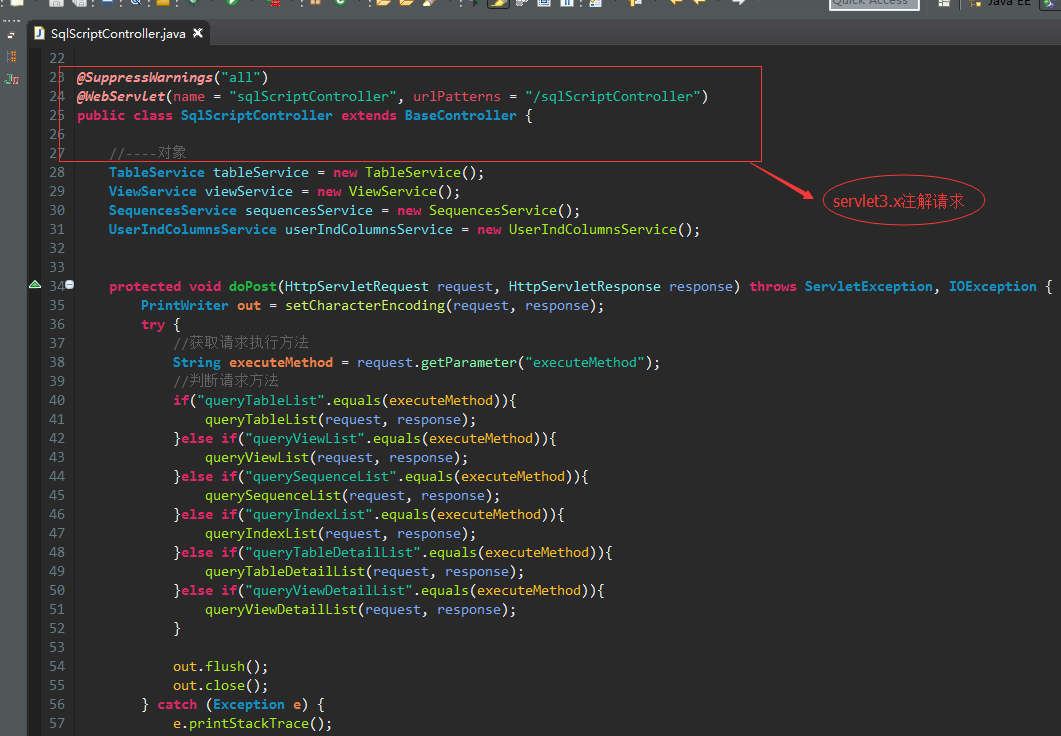
### 6.1.2Service



### 6.1.3Util

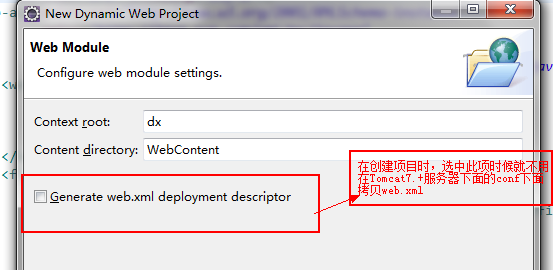


### 6.1.4Controler

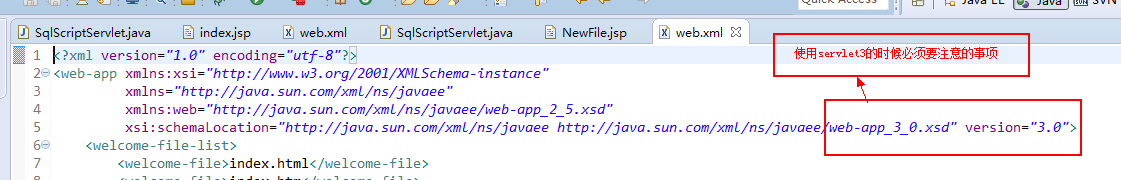


### 6.1.5其他说明

1. Servlet3.x注意事项1



1. Servlet3.x注意事项2

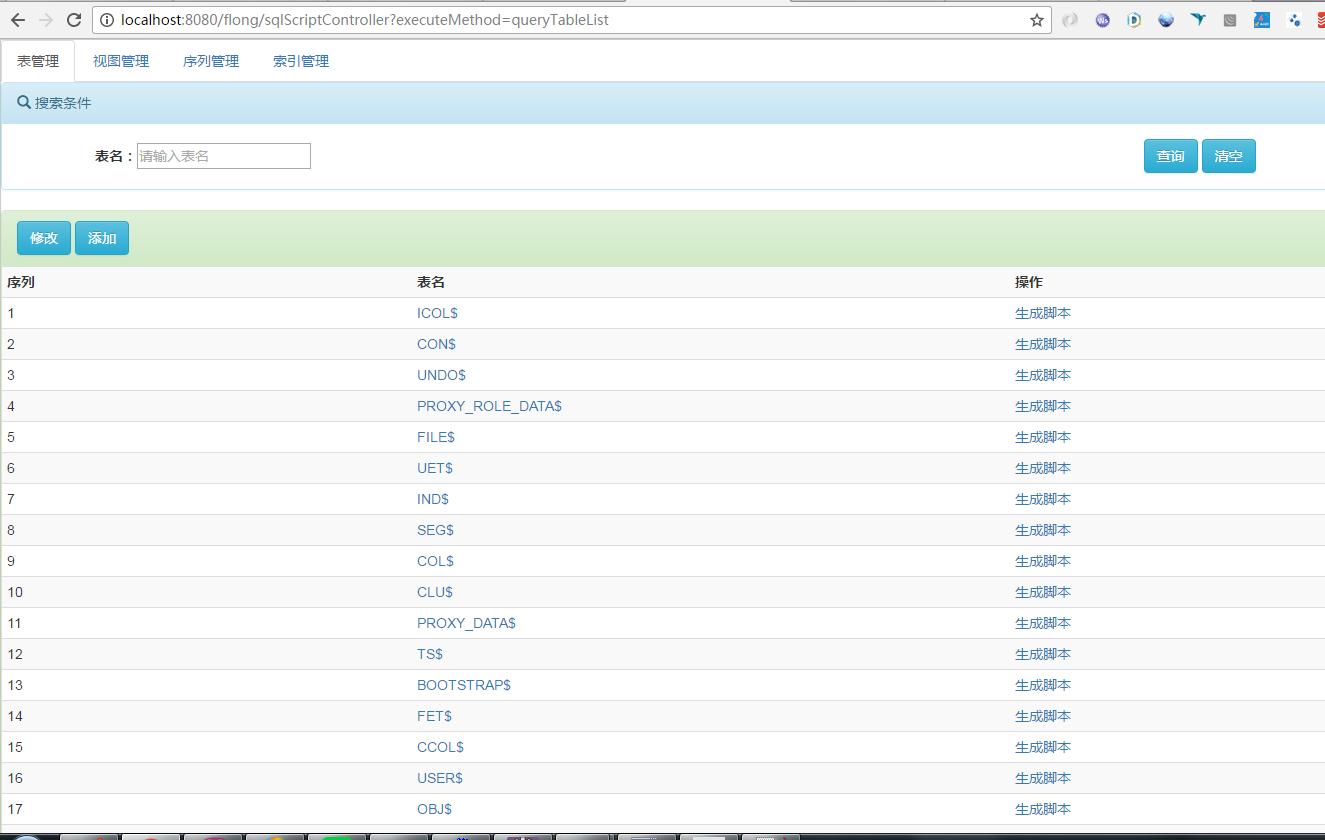


由于jdbc实现代码较多和关闭数据源处理不好经常断，后面所有功能转为spring架构实现

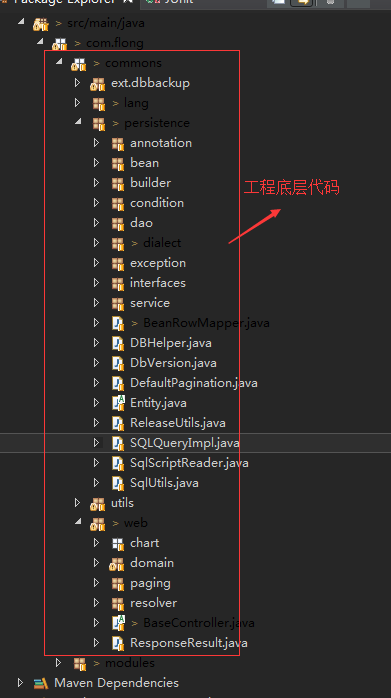
### 6.1.6效果

启动Tomcat访问url：

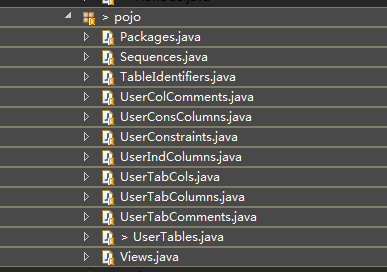
<http://localhost:8080/flong/sqlScriptController?executeMethod=queryTableList>



## 6.2SpringMVC+JdbcTemplate实现



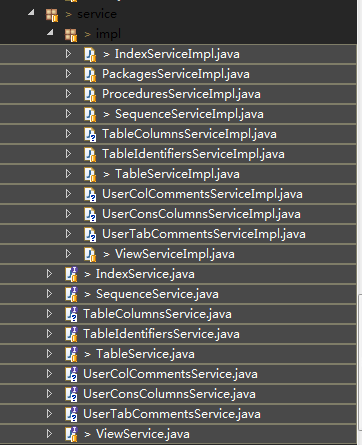
### 6.2.1Pojo



|  |  |
| --- | --- |
| Packages.java | 数据库包实体 |
| Sequences.java | 序列实体 |
| TableIdentifiers.java | 唯一主键表实体 |
| UserColComments.java | 用户列描述表实体 |
| UserConsColumns.java | 用户主键约束表实体 |
| UserConstraints.java | 用户外键约束表实体 |
| UserIndColumns.java | 用户索引表实体 |
| UserTabCols.java | 用户表与列表实体 |
| UserTabColumns.java | 用户表与列表实体 |
| UserTabComments.java | 用户表描述表实体 |
| UserTables.java | 用户表实体 |
| Views.java | 视图表实体 |

|  |
| --- |
| **package** com.flong.modules.pojo;  **import** java.io.Serializable;  **import** java.math.BigDecimal;  **import** java.util.Date;  **import** com.flong.commons.persistence.Entity;  **import** com.flong.commons.persistence.annotation.Column;  **import** com.flong.commons.persistence.annotation.Id;  **import** com.flong.commons.persistence.annotation.Relation;  **import** com.flong.commons.utils.Table;  /\*\*  \*  \* **@Description** 系统用户表  \* **@ClassName** UserTables  \* **@Date** 2017年2月28日 下午5:54:26  \* **@Author** liangjilong  \* **@Copyright** (c) All Right Reserved ; 2017.  \*/  @SuppressWarnings("all")  @Relation(TableIdentifiers.***TABLE***)  **public** **class** UserTables **extends** Entity **implements** Serializable{  /\*\* 表名常量 \*/  **public** **static** **final** String ***TABLE*** = Table.***USER\_TABLES***;  //ID和@Column必须配置一个jdbcTemplate底层代码控制，如果需要配置所有字段jpa可以用jdbcTemplate封装的  //saveOrUpdate,具体代码请阅读TableIdentifiersDAOImpl  @Id  @Column("table\_name")  **private** String table\_name;  **private** String tablespace\_name;  **private** String cluster\_name;  **private** String iot\_name;  **private** String status;  **private** BigDecimal pct\_free;  **private** BigDecimal pct\_used;  **private** BigDecimal ini\_trans;  **private** BigDecimal max\_trans;  **private** BigDecimal initial\_extent;  **private** BigDecimal next\_extent;  **private** BigDecimal min\_extents;  **private** BigDecimal max\_extents;  **private** String pct\_increase;  **private** BigDecimal freelists;  **private** BigDecimal freelist\_groups;  **private** String logging;  **private** String backed\_up;  **private** BigDecimal num\_rows;  **private** BigDecimal blocks;  **private** BigDecimal empty\_blocks;  **private** BigDecimal avg\_space;  **private** BigDecimal chain\_cnt;  **private** BigDecimal avg\_row\_len;  **private** BigDecimal avg\_space\_freelist\_blocks;  **private** BigDecimal num\_freelist\_blocks;  **private** String degree;  **private** String instances;  **private** String cache;  **private** String table\_lock;  **private** BigDecimal sample\_size;  **private** Date last\_analyzed;  **private** String partitioned;  **private** String iot\_type;  **private** String temporary;  **private** String secondary;  **private** String nested;  **private** String buffer\_pool;  **private** String flash\_cache;  **private** String cell\_flash\_cache;  **private** String row\_movement;  **private** String global\_stats;  **private** String user\_stats;  **private** String duration;  **private** String skip\_corrupt;  **private** String monitoring;  **private** String cluster\_owner;  **private** String dependencies;  **private** String compression;  **private** String compress\_for;  **private** String dropped;  **private** String read\_only;  **private** String segment\_created;  **private** String result\_cache;  } |

### 6.2.2Service

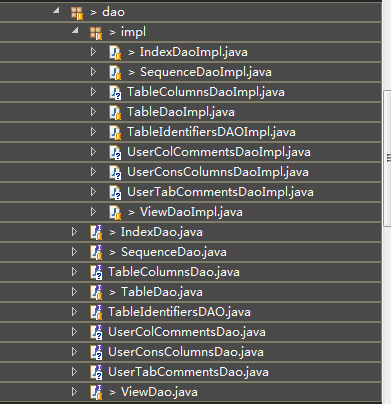


|  |
| --- |
| **public** **interface** TableService **extends** EntityDao<UserTables> {    /\*\*  \* **@Description** 查询所有表信息包含分页  \* **@Author** liangjilong  \* **@Date** 2017年3月28日 下午4:43:34  \* **@param** page  \* **@param** object  \* **@return** 参数  \* **@return** List<UserTables> 返回类型  \*/  **public** List<UserTables> **queryTables**(**SimplePage** page, **UserTables** object) ;    /\*\*  \* **@Description** 根据表名查询表信息  \* **@Author** liangjilong  \* **@Date** 2017年3月28日 下午4:43:57  \* **@param** tableName  \* **@return** 参数  \* **@return** List<UserTables> 返回类型  \*/  **public** List<UserTables> **queryTableByName**(**String** tableName);    /\*\*  \* **@Description** 根据表名进行生成数据库存在表的创建表的SQL脚本，如果是页面传多个表的时候可以遍历此方法  \* **@Author** liangjilong  \* **@Date** 2017年3月28日 下午4:44:15  \* **@param** tableName  \* **@return** 参数  \* **@return** String 返回类型  \*/  **public** **String** **createTableSQLScript**(**String** tableName);  /\*\*  \* **@Description** 根据表名进行生成数据库存在表的删除表的的SQL脚本，如果是页面传多个表的时候可以遍历此方法  \* **@Author** liangjilong  \* **@Date** 2017年3月28日 下午4:55:02  \* **@param** table\_name  \* **@return** 参数  \* **@return** String 返回类型  \*/  **public** **String** **dropTableSQLScript**(**String** table\_name); |

脚本组装

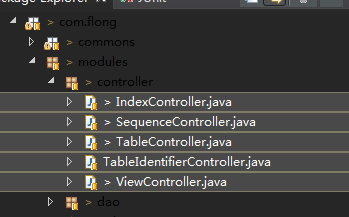
|  |
| --- |
| package com.flong.modules.service.impl;  import java.math.BigDecimal;  import java.util.List;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Service;  import org.springframework.transaction.annotation.Transactional;  import com.flong.commons.persistence.bean.SimplePage;  import com.flong.commons.persistence.dao.impl.EntityDaoSupport;  import com.flong.commons.utils.ObjectUtil;  import com.flong.modules.dao.TableColumnsDao;  import com.flong.modules.dao.TableDao;  import com.flong.modules.dao.UserColCommentsDao;  import com.flong.modules.dao.UserConsColumnsDao;  import com.flong.modules.dao.UserTabCommentsDao;  import com.flong.modules.pojo.Sequences;  import com.flong.modules.pojo.UserColComments;  import com.flong.modules.pojo.UserConsColumns;  import com.flong.modules.pojo.UserTabColumns;  import com.flong.modules.pojo.UserTabComments;  import com.flong.modules.pojo.UserTables;  import com.flong.modules.service.TableService;  /\*\*  \* @Description TODO  \* @ClassName TableService  \* @Date 2017年3月1日 上午10:48:44  \* @Author liangjilong  \* @Copyright (c) All Right Reserved , 2017.  \*/  @Service  @Transactional  @SuppressWarnings("all")  public class TableServiceImpl extends EntityDaoSupport<UserTables> implements TableService {  @Autowired TableDao tableDao;  @Autowired TableColumnsDao tableColumnsDao;  @Autowired UserColCommentsDao userColCommentsDao;  @Autowired UserTabCommentsDao userTabCommentsDao;  @Autowired UserConsColumnsDao userConsColumnsDao;    public List<UserTables> queryTables(SimplePage page, UserTables object) {  return tableDao.queryTables(page,object);  }  public List<UserTables> queryTableByName(String table\_name){  return tableDao.queryTableByName(table\_name);  }  @Override  public String createTableSQLScript(String tableName) {  String result = "";  //1、判断是否存在这个表  List<UserTables> findList = queryTableByName( tableName);  //2、组装execute immediate 'create table ...'  if(findList!=null &&findList.size()>0){  StringBuffer buffer = new StringBuffer();  //3、获取表的字段名称，并且获取约束是否为空(Null or Not null)  List<UserTabColumns> userTabColumnList = tableColumnsDao.getUserTabColumnsByName(tableName);  List<UserTabComments> userTabCommentList = userTabCommentsDao.getUserTabCommentsByName(tableName);  List<UserColComments> userColCommentList = userColCommentsDao.getUserColCommentsByName(tableName);  UserConsColumns userConsColumns = userConsColumnsDao.getUserConsColumnsByName(tableName);  //----组装表字段列名  int size = userTabColumnList.size()-1;//计算最后一个表列名  buffer.append("--创建"+tableName+"表").append(ENTER);  buffer.append("declare").append(ENTER);  buffer.append(TAB).append("iCnt number := 0;").append(ENTER);  buffer.append("begin ").append(ENTER);  buffer.append(" select count(\*) into iCnt from user\_tables where lower(table\_name) = lower('"+tableName+"');").append(ENTER);  buffer.append(" if iCnt = 0 then ").append(ENTER);  buffer.append(TAB).append("execute immediate 'create table "+tableName+"").append(ENTER);  buffer.append(TAB).append("(").append(ENTER);    for (int i = 0; i < userTabColumnList.size(); i++) {  UserTabColumns userTabColumns = userTabColumnList.get(i);  UserColComments userColComments = userColCommentList.get(i);    String column\_name1 = userColComments.getColumn\_name();//列名    String column\_name2 = userTabColumns.getColumn\_name();//列名  String nullable = userTabColumns.getNullable();//是否允许为空  String data\_type = userTabColumns.getData\_type();//数据库表的字段类型.  BigDecimal data\_length = userTabColumns.getData\_length();  BigDecimal data\_precision = userTabColumns.getData\_precision();  BigDecimal data\_scale = userTabColumns.getData\_scale();    if(ObjectUtil.isNotEmpty(column\_name1)&& ObjectUtil.isNotEmpty(column\_name2) && column\_name1.equalsIgnoreCase(column\_name2)){    //判断表字段类型是否为NULL或NOT NULL  if(ObjectUtil.isNotEmpty(nullable) && !"N".equalsIgnoreCase(nullable)){  if(size==i){  if("VARCHAR2".equalsIgnoreCase(data\_type) || "NVARCHAR2".equalsIgnoreCase(data\_type) ||"CHAR".equalsIgnoreCase(data\_type)){  buffer.append(TAB+TAB+column\_name1+TAB+data\_type+"("+data\_length+")" ).append(ENTER);  }else if("NUMBER".equalsIgnoreCase(data\_type)){  if(data\_precision!= null && data\_scale!=null){  buffer.append(TAB+TAB+column\_name1+TAB+data\_type+"("+data\_precision+","+data\_scale+") ").append(ENTER);  }else if(data\_precision!= null && data\_scale==null){  buffer.append(TAB+TAB+column\_name1+TAB+data\_type+"("+data\_precision+",0) ").append(ENTER);  }else{  buffer.append(TAB+TAB+column\_name1+TAB+data\_type).append(ENTER);  }  }else if(("DATE".equalsIgnoreCase(data\_type))||(data\_type!=null && data\_type.contains("TIMESTAMP")) ){  buffer.append(TAB+TAB+column\_name1+TAB+data\_type).append(ENTER);  }else if("LONG".equalsIgnoreCase(data\_type) ||"NCLOB".equalsIgnoreCase(data\_type)){  buffer.append(TAB+TAB+column\_name1+TAB+data\_type).append(ENTER);  }  }else{  //为空但是有逗号分开  if("VARCHAR2".equalsIgnoreCase(data\_type) || "NVARCHAR2".equalsIgnoreCase(data\_type) ||"CHAR".equalsIgnoreCase(data\_type)){  buffer.append(TAB+TAB+column\_name1+TAB+data\_type+"("+data\_length+")" ).append(COMMA).append(ENTER);  }else if("NUMBER".equalsIgnoreCase(data\_type)){  if(data\_precision!= null && data\_scale!=null){  buffer.append(TAB+TAB+column\_name1+TAB+data\_type+"("+data\_precision+","+data\_scale+") ").append(COMMA).append(ENTER);  }else if(data\_precision!= null && data\_scale==null){  buffer.append(TAB+TAB+column\_name1+TAB+data\_type+"("+data\_precision+",0) ").append(COMMA).append(ENTER);  }else{  buffer.append(TAB+TAB+column\_name1+TAB+data\_type).append(COMMA).append(ENTER);  }  }else if(("DATE".equalsIgnoreCase(data\_type))||(data\_type!=null && data\_type.contains("TIMESTAMP")) ){  buffer.append(TAB+TAB+column\_name1+TAB+data\_type).append(COMMA).append(ENTER);  }else if("LONG".equalsIgnoreCase(data\_type) ||"NCLOB".equalsIgnoreCase(data\_type)){  buffer.append(TAB+TAB+column\_name1+TAB+data\_type).append(COMMA).append(ENTER);  }  }  }else{  if(size==i){  if("VARCHAR2".equalsIgnoreCase(data\_type) || "NVARCHAR2".equalsIgnoreCase(data\_type) ||"CHAR".equalsIgnoreCase(data\_type)){  buffer.append(TAB+TAB+column\_name1+TAB+data\_type+"("+data\_length+") not null " ).append(ENTER);  }else if("NUMBER".equalsIgnoreCase(data\_type)){  if(data\_precision!= null && data\_scale!=null){  buffer.append(TAB+TAB+column\_name1+TAB+data\_type+"("+data\_precision+","+data\_scale+") not null").append(ENTER);  }else if(data\_precision!= null && data\_scale==null){  buffer.append(TAB+TAB+column\_name1+TAB+data\_type+"("+data\_precision+",0) not null ").append(ENTER);  }else{  buffer.append(TAB+TAB+column\_name1+TAB+data\_type+"").append(ENTER);  }  }else if(("DATE".equalsIgnoreCase(data\_type))||(data\_type!=null && data\_type.contains("TIMESTAMP")) ){  buffer.append(TAB+TAB+column\_name1+TAB+data\_type).append(ENTER);  }else if("LONG".equalsIgnoreCase(data\_type) ||"NCLOB".equalsIgnoreCase(data\_type)){  buffer.append(TAB+TAB+column\_name1+TAB+data\_type).append(ENTER);  }    }else{  if("VARCHAR2".equalsIgnoreCase(data\_type) || "NVARCHAR2".equalsIgnoreCase(data\_type) ||"CHAR".equalsIgnoreCase(data\_type)){  buffer.append(TAB+TAB+column\_name1+TAB+data\_type+"("+data\_length+") not null " ).append(COMMA).append(ENTER);  }else if("NUMBER".equalsIgnoreCase(data\_type)){  if(data\_precision!= null && data\_scale!=null){  buffer.append(TAB+TAB+column\_name1+TAB+data\_type+"("+data\_precision+","+data\_scale+") not null").append(COMMA).append(ENTER);  }else if(data\_precision!= null && data\_scale==null){  buffer.append(TAB+TAB+column\_name1+TAB+data\_type+"("+data\_precision+",0) not null ").append(COMMA).append(ENTER);  }else{  buffer.append(TAB+TAB+column\_name1+TAB+data\_type+" ").append(COMMA).append(ENTER);  }  }else if(("DATE".equalsIgnoreCase(data\_type))||(data\_type!=null && data\_type.contains("TIMESTAMP")) ){  buffer.append(TAB+TAB+column\_name1+TAB+data\_type).append(COMMA).append(ENTER);  }else if("LONG".equalsIgnoreCase(data\_type) ||"NCLOB".equalsIgnoreCase(data\_type)){  buffer.append(TAB+TAB+column\_name1+TAB+data\_type).append(COMMA).append(ENTER);  }  }  }  }  }  buffer.append(TAB).append(")';").append(ENTER);  //4、获取表名给予表加备注.（有就加，无默认表描述）  if(ObjectUtil.isNotEmpty(userTabCommentList) && ObjectUtil.isNotEmpty(userTabCommentList.get(0))){  String comments = userTabCommentList.get(0).getComments();  comments = (comments==null?" this is "+tableName+" tableName ":comments);  buffer.append(TAB).append("execute immediate 'comment on table "+tableName +" is ''"+comments+"'''").append(";").append(ENTER);  }  //5、添加主键执行SQL脚本  if(ObjectUtil.isNotEmpty(userConsColumns) && ObjectUtil.isNotEmpty(userConsColumns.getConstraint\_name())){  buffer.append(TAB).append("execute immediate 'alter table "+tableName).append(" add constraint "+userConsColumns.getConstraint\_name()+" ");  buffer.append("primary key ("+userConsColumns.getColumn\_name()+")' ;").append(ENTER);  }  //6、获取表字段名给予字段加备注（有就加，无默认字段描述）  for(UserColComments col: userColCommentList){  String comments = col.getComments();  String column\_name = col.getColumn\_name();  comments = (comments==null?" this is "+column\_name+" columnName ":comments);  buffer.append(TAB).append("execute immediate 'comment on column "+tableName+"."+col.getColumn\_name()+" is ''"+comments+"'''").append(";").append(ENTER);  }  //---end if  buffer.append(" end if;").append(ENTER);  buffer.append("end;").append(ENTER);  buffer.append("/").append(ENTER);  result = buffer.toString();  }  return result;  }    @Override  public String dropTableSQLScript(String table\_name) {    List<UserTables> list = queryTableByName(table\_name);  StringBuffer buffer = new StringBuffer();  if(list!=null && list.size()>0){  UserTables sequences = list.get(0);  buffer.append("--"+table\_name).append(ENTER);  buffer.append("declare").append(ENTER);  buffer.append(TAB+TAB).append("iCnt number := 0;").append(ENTER);  buffer.append(TAB).append("begin").append(ENTER);  buffer.append(TAB).append(" select count(\*) into iCnt from user\_tables where lower(table\_name) = '"+table\_name+"'; --根据表进行查询").append(ENTER);  buffer.append(TAB).append(" if iCnt > 0 then ").append(ENTER);  buffer.append(TAB).append(" execute immediate 'drop table "+table\_name+"'; --删除表,表结构和数据一起清空 ").append(ENTER);  buffer.append(TAB).append(" end if;").append(ENTER);  buffer.append(TAB).append("end;").append(ENTER);  buffer.append("/").append(ENTER);  return buffer.toString();  }  return null;  }  } |

### 6.2.3DAO



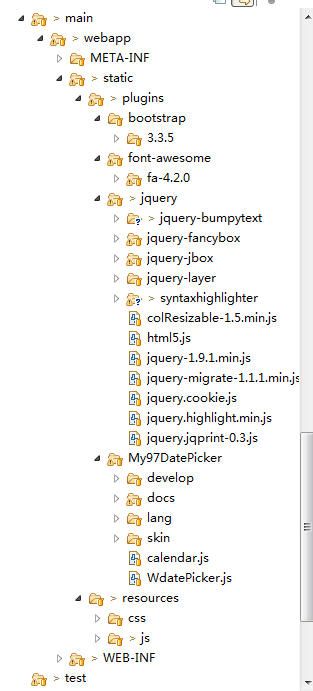
|  |
| --- |
| **public** **interface** TableDao **extends** EntityDao<UserTables> {    /\*\*  \* **@Description** 根据表名查询表所有信息，包含分页  \* **@Author** liangjilong  \* **@Date** 2017年3月28日 下午4:53:01  \* **@param** tableName  \* **@return** 参数  \* **@return** List<UserTables> 返回类型  \*/  **public** List<UserTables> **queryTables**(**SimplePage** page, **UserTables** object);    /\*\*  \* **@Description** 根据表名查询表信息  \* **@Author** liangjilong  \* **@Date** 2017年3月28日 下午4:53:01  \* **@param** tableName  \* **@return** 参数  \* **@return** List<UserTables> 返回类型  \*/  **public** List<UserTables> **queryTableByName**(**String** tableName);      /\*\*  \* **@Description** 根据表名创建表SQL脚本  \* **@Author** liangjilong  \* **@Date** 2017年3月28日 下午4:52:29  \* **@param** tableName  \* **@return** 参数  \* **@return** String 返回类型  \*/  **public** **String** **createTableSQLScript**(**String** tableName);    } |

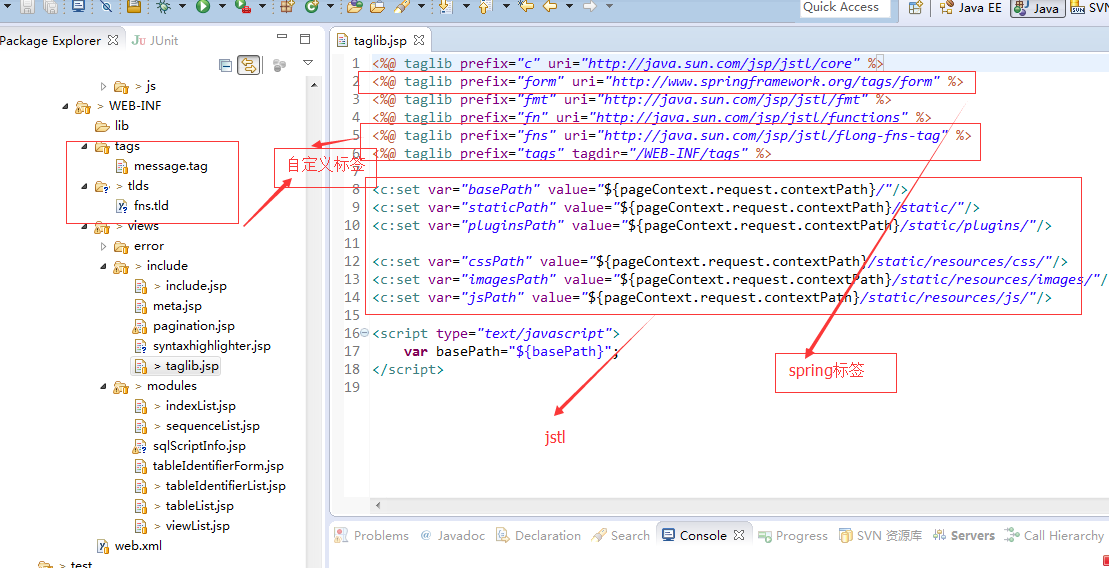
### 6.2.4Controler



|  |
| --- |
| package com.flong.modules.controller;  import java.util.List;  import javax.servlet.http.HttpServletRequest;  import javax.servlet.http.HttpServletResponse;  import org.springframework.beans.factory.annotation.Autowired;  import org.springframework.stereotype.Controller;  import org.springframework.web.bind.annotation.PathVariable;  import org.springframework.web.bind.annotation.RequestMapping;  import com.alibaba.fastjson.JSONObject;  import com.flong.codegenerator.FileUtils;  import com.flong.commons.persistence.bean.SimplePage;  import com.flong.commons.web.BaseController;  import com.flong.modules.pojo.UserTables;  import com.flong.modules.service.TableService;  /\*\*  \* @author liangjilong  \*  \*/  @Controller  @RequestMapping("table")  public class TableController extends BaseController{  @Autowired TableService tableService;      /\*\*\*  \* 查询所有Oracle内置数据库表或者视图信息  \* @param tableIdentifiers  \* @param page  \* @param request  \* @param response  \* @return  \*/  @RequestMapping(value="list")  public String list(UserTables userTables,SimplePage page ,HttpServletRequest request ){  List<UserTables> findList = tableService.queryTables(page, userTables);  request.setAttribute("userTables", userTables);  request.setAttribute("page", page);  request.setAttribute("list", findList);  return "modules/tableList";  }    /\*\*  \* @Description 根据表名进行查询表信息  \* @Author liangjilong  \* @Date 2017年3月27日 上午8:23:14  \* @param table\_name  \* @param request  \* @return 参数  \* @return String 返回类型  \*/  @RequestMapping(value="createTableSQLScript/{table\_name}")  public String createTableSQLScript(@PathVariable String table\_name,HttpServletRequest request ){  String resultSQLStr = tableService.createTableSQLScript(table\_name);  request.setAttribute("resultSQLStr", resultSQLStr);  return "modules/sqlScriptInfo";  }    /\*\*  \* @Description 根据表名进行查询表信息  \* @Author liangjilong  \* @Date 2017年3月27日 上午8:23:14  \* @param table\_name  \* @param request  \* @return 参数  \* @return String 返回类型  \*/  @RequestMapping(value="dropTableSQLScript/{table\_name}")  public String dropTableSQLScript(@PathVariable String table\_name,HttpServletRequest request ){  String resultSQLStr = tableService.dropTableSQLScript(table\_name);  request.setAttribute("resultSQLStr", resultSQLStr);  return "modules/sqlScriptInfo";  }    /\*\*  \*  \* @Description 导出SQL  \* 本来html的textarea标签可以使用js实现拷贝的，为了模仿iteye或者csdn博客开发页面拷贝功能的，由于找了好几个插件都没试出来，  \* 而且他们都是使用swf插件实现，在脚本高亮在显示关键字就研究了SyntaxHighlighter导致textarea使用js拷贝功能失效就放弃按钮拷  \* 贝功能。为了更加方便通过文件流处理，指定电脑硬盘路径和名称，并以.sql后缀文件直接输出，这样更方便帮助我们直接省了Ctrl+C  \* 和Ctrl+V内容， //再新建文件整理脚本工作动作。cnblogs博客是双击代码，按 Ctrl+C复制代码（此时是jQuery高亮插件自带功能提示）  \* 参考博客：http://www.cnblogs.com/tylerdonet/p/4533782.html  \* @Author liangjilong  \* @Date 2017年3月27日 上午8:21:15  \* @param table\_name  \* @param request  \* 参数  \* @return void 返回类型  \*/  @RequestMapping(value="exportCreateSQL/{table\_name}")  public void exportCreateSQL(@PathVariable String table\_name,HttpServletResponse response ){  String resultSQLStr = tableService.createTableSQLScript(table\_name);  //这里为了快速方便使用指定路径实现，并以.sql后缀文件导出到指定硬盘。可以根据自己的情况进行修改，比如：你可以使用SpringMVC下载文件方式实现，  String path = "C://output-sql/"+"TableSQL-"+formatDate+".sql";  FileUtils.save(path, resultSQLStr);//保存文件  JSONObject jsonObject = new JSONObject();//使用fastjson技术实现ajax异步处理  jsonObject.put("path", path);  jsonObject.put("result", "1");  this.writeJson(jsonObject, response);  }    /\*\*  \*  \* @Description 导出SQL  \* 本来html的textarea标签可以使用js实现拷贝的，为了模仿iteye或者csdn博客开发页面拷贝功能的，由于找了好几个插件都没试出来，  \* 而且他们都是使用swf插件实现，在脚本高亮在显示关键字就研究了SyntaxHighlighter导致textarea使用js拷贝功能失效就放弃按钮拷  \* 贝功能。为了更加方便通过文件流处理，指定电脑硬盘路径和名称，并以.sql后缀文件直接输出，这样更方便帮助我们直接省了Ctrl+C  \* 和Ctrl+V内容， //再新建文件整理脚本工作动作。cnblogs博客是双击代码，按 Ctrl+C复制代码（此时是jQuery高亮插件自带功能提示）  \* 参考博客：http://www.cnblogs.com/tylerdonet/p/4533782.html  \* @Author liangjilong  \* @Date 2017年3月27日 上午8:21:15  \* @param table\_name  \* @param request  \* 参数  \* @return void 返回类型  \*/  @RequestMapping(value="exportDropSQL/{table\_name}")  public void exportDropSQL(@PathVariable String table\_name,HttpServletResponse response ){  String resultSQLStr = tableService.dropTableSQLScript(table\_name);  //这里为了快速方便使用指定路径实现，并以.sql后缀文件导出到指定硬盘。可以根据自己的情况进行修改，比如：你可以使用SpringMVC下载文件方式实现，  String path = "C://output-sql/"+"TableSQL-"+formatDate+".sql";  FileUtils.save(path, resultSQLStr);//保存文件  JSONObject jsonObject = new JSONObject();//使用fastjson技术实现ajax异步处理  jsonObject.put("path", path);  jsonObject.put("result", "1");  this.writeJson(jsonObject, response);    }  } |

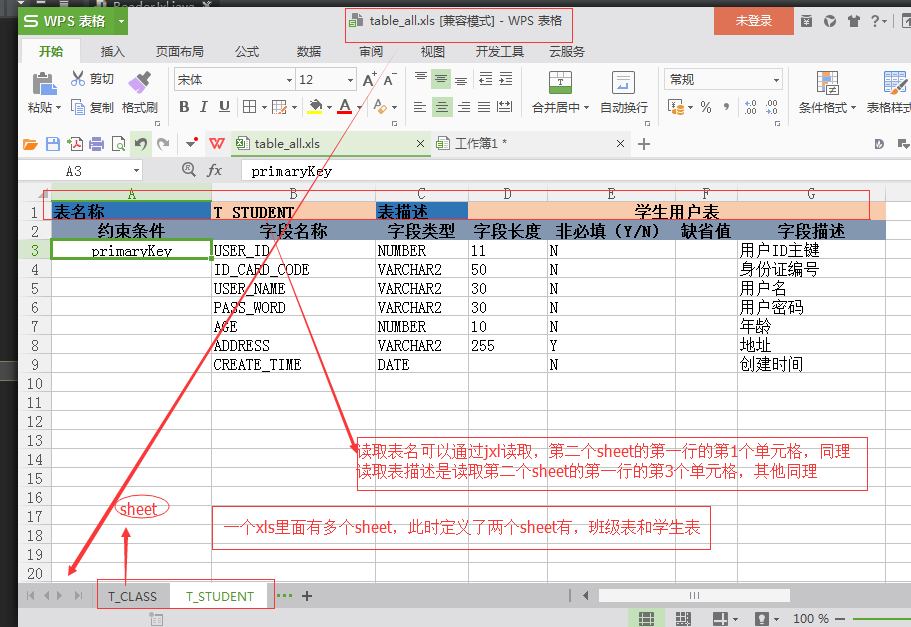
### 6.2.5前端技术





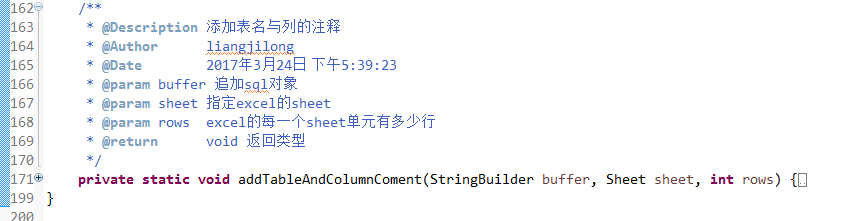
### 6.2.6JXL实现

Excel模板

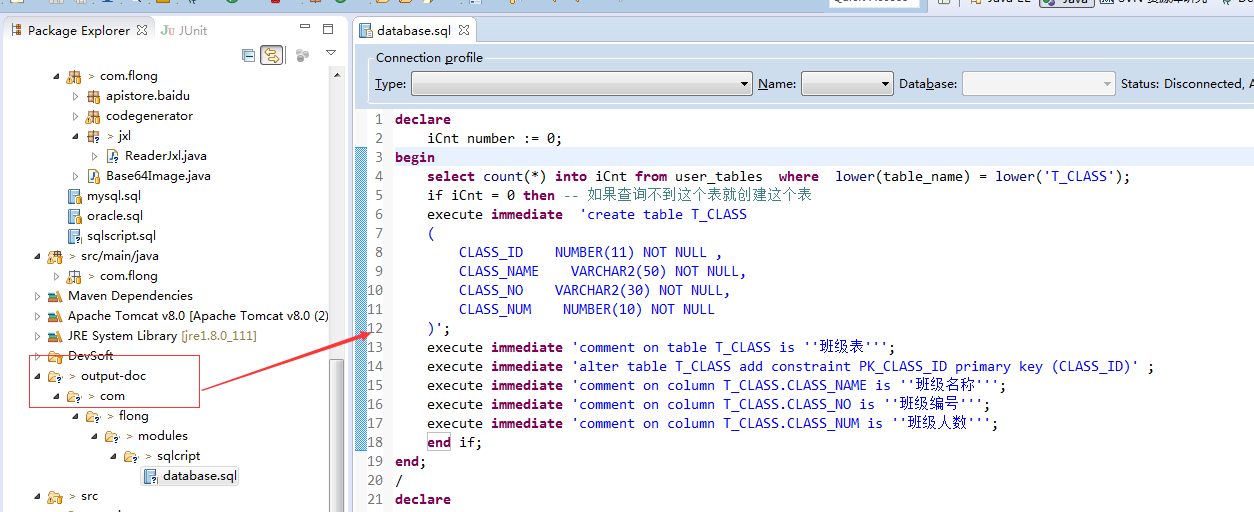


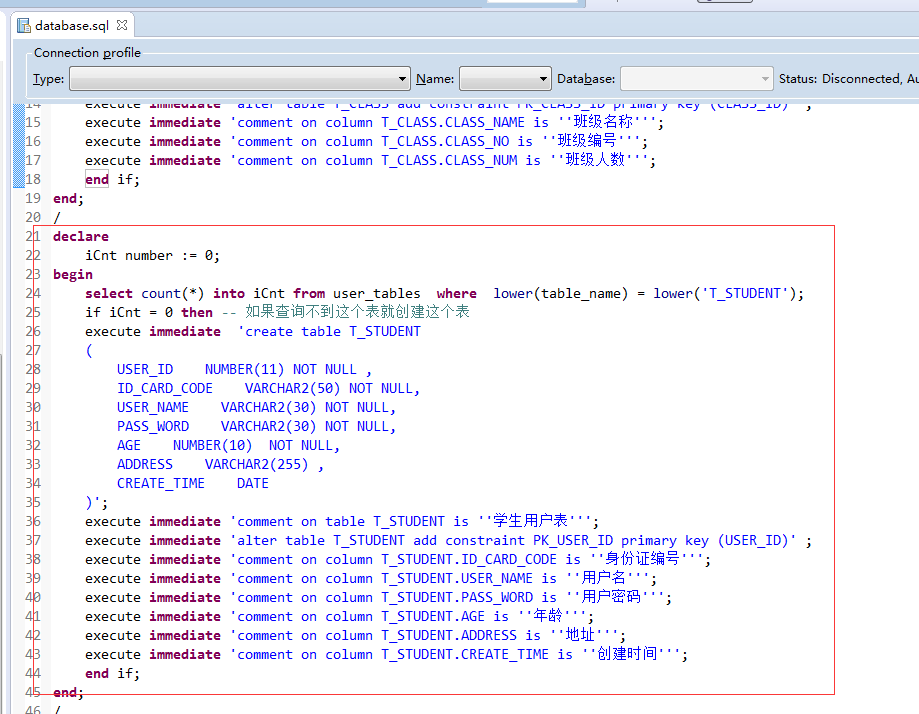
运行代码：



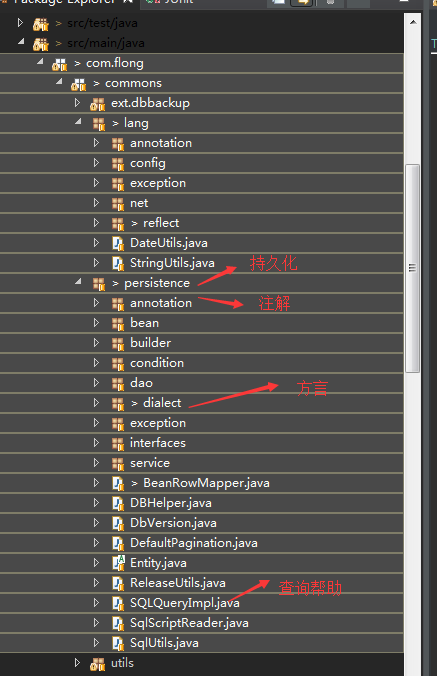


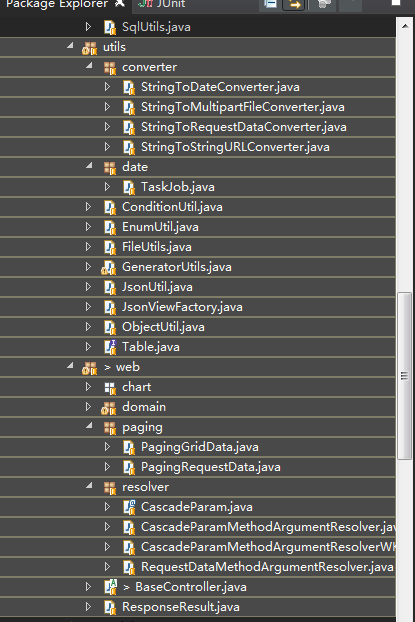
生成声明式高级SQL脚本



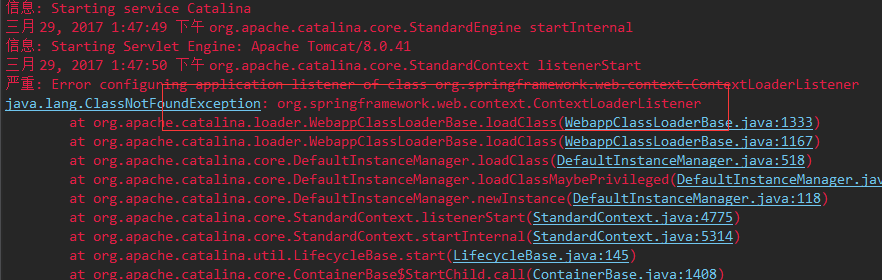


### 6.2.7Util底层封装代码

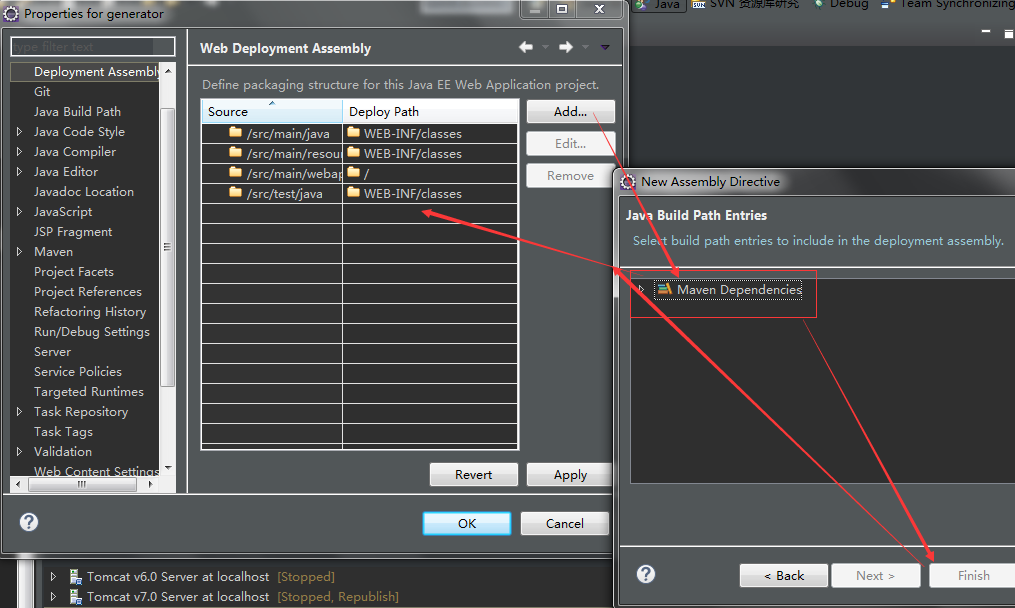


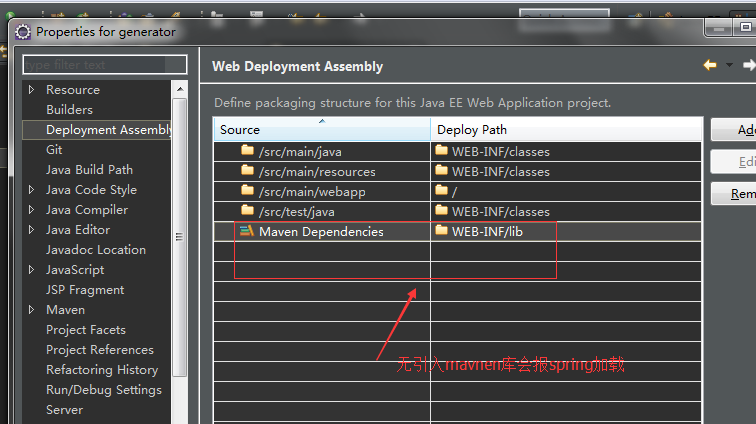


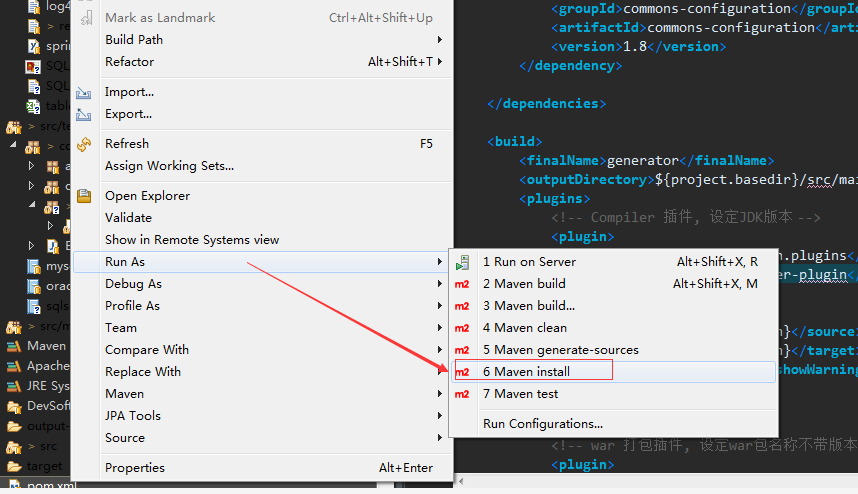
### 6.2.8其他说明

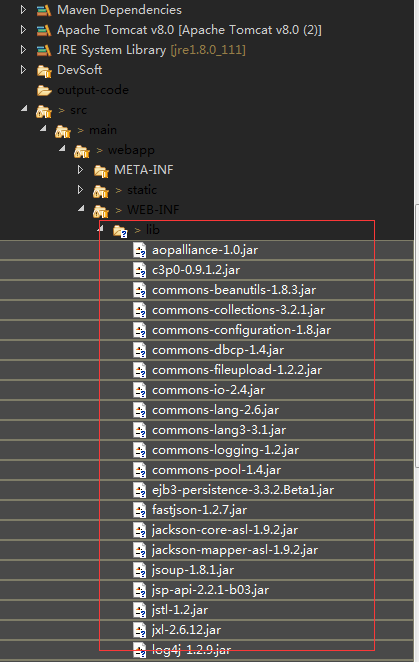
如果在运行项目的时候，报Spring监听的时候，说明没找到Maven，引用jar，

解决方法加上Maven 库或者运行maven istanll拷贝jar到工程lib下面【在pom.xml已经配置好】拷贝成功之后此时可以再运行web工程就无错误.



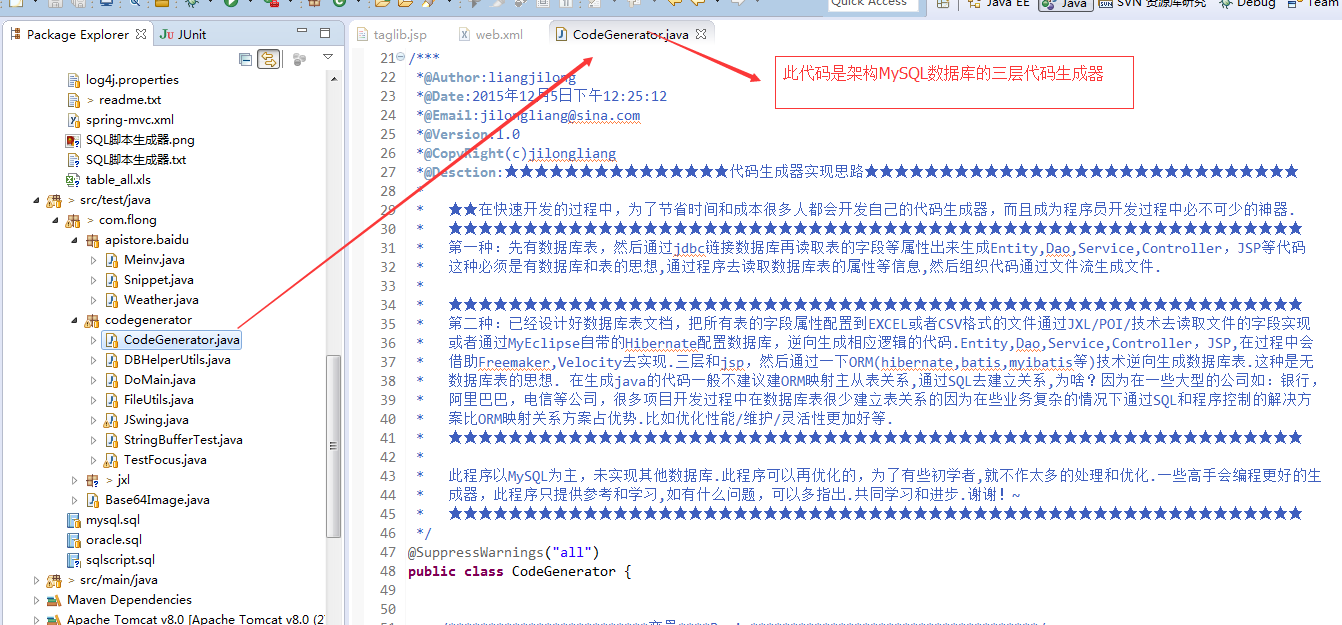






Web.xml配置

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns=*"http://java.sun.com/xml/ns/javaee"* xmlns:web=*"http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"* xsi:schemaLocation=*"http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"* version=*"2.5"*>  <display-name>flong</display-name>    <!-- 参数初始化类配置 -->  <context-param>  <param-name>contextConfigLocation</param-name>  <param-value>  classpath\*:spring/spring-context.xml,  classpath\*:spring/spring-timer.xml  </param-value>  </context-param>  <context-param>  <param-name>spring.profiles.default</param-name>  <param-value>production</param-value>  </context-param>    <!-- ===================================encoding=========================================== -->  <listener>  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>  </listener>  <filter>  <filter-name>encodingFilter</filter-name>  <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>  <init-param>  <param-name>encoding</param-name>  <param-value>UTF-8</param-value>  </init-param>  <init-param>  <param-name>forceEncoding</param-name>  <param-value>true</param-value>  </init-param>  </filter>  <filter-mapping>  <filter-name>encodingFilter</filter-name>  <url-pattern>/\*</url-pattern>  </filter-mapping>    <!-- ===================================SpringMVC=========================================== -->  <servlet>  <servlet-name>SpringMVC</servlet-name>  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  <init-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath\*:spring-mvc.xml</param-value>  </init-param>  <load-on-startup>1</load-on-startup>  </servlet>  <servlet-mapping>  <servlet-name>SpringMVC</servlet-name>  <url-pattern>/</url-pattern>  </servlet-mapping>  <!-- ===================================error-page=========================================== -->  <error-page>  <exception-type>java.lang.Throwable</exception-type>  <location>/WEB-INF/views/error/500.jsp</location>  </error-page>  <error-page>  <error-code>500</error-code>  <location>/WEB-INF/views/error/500.jsp</location>  </error-page>  <error-page>  <error-code>400</error-code>  <location>/WEB-INF/views/error/400.jsp</location>  </error-page>  <error-page>  <error-code>403</error-code>  <location>/WEB-INF/views/error/403.jsp</location>  </error-page>  <error-page>  <error-code>404</error-code>  <location>/WEB-INF/views/error/404.jsp</location>  </error-page>  </web-app> |

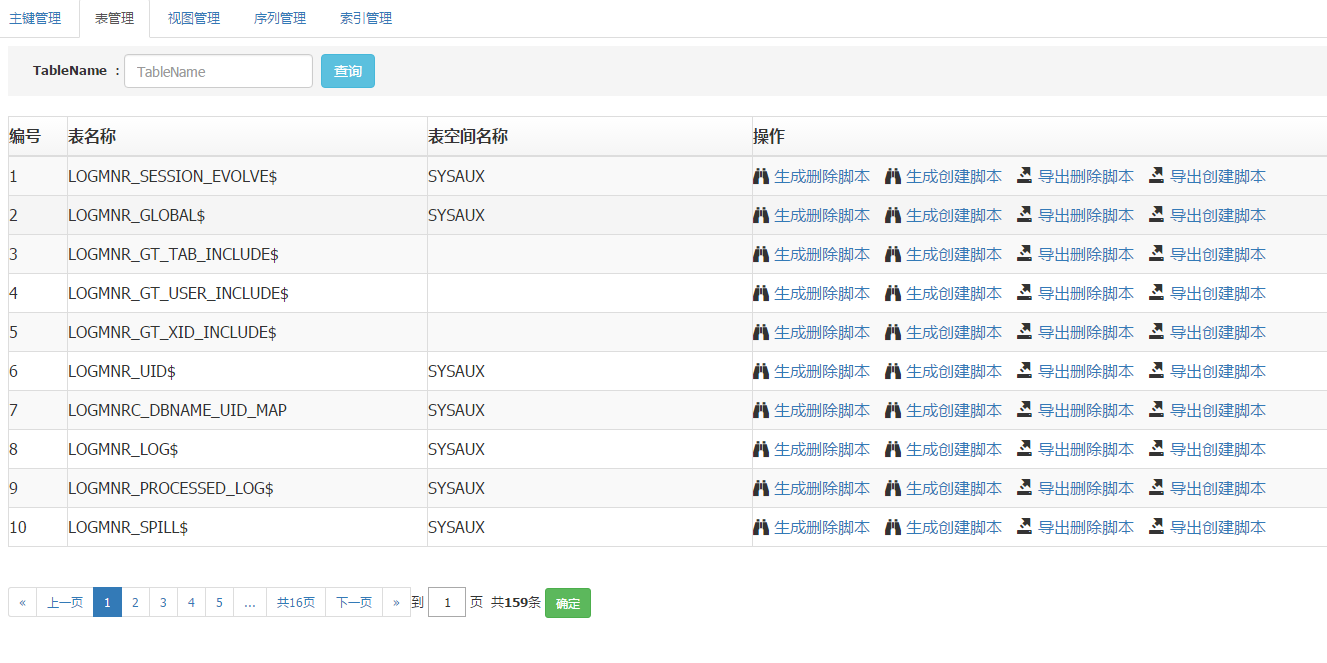


### 6.2.9功能与效果

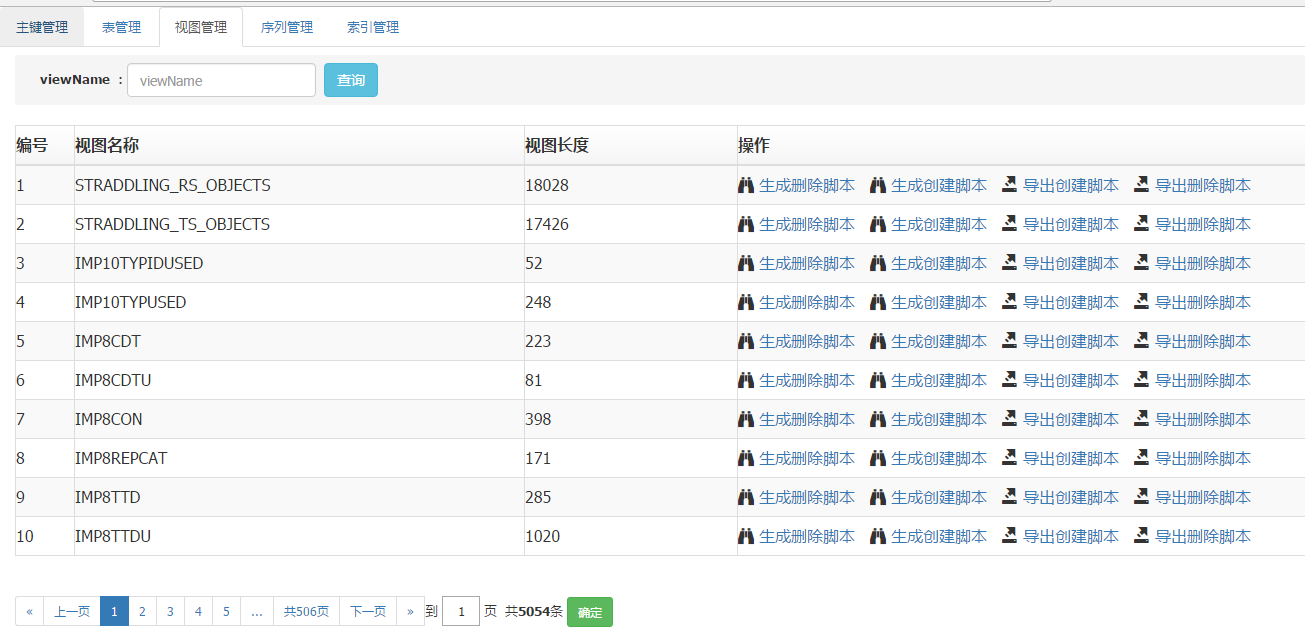
其中有表管理，视图管理，序列管理，索引管理，每个管理都有【删除与创建脚本功能，导出删除和导出创建脚本功能】

启动Tomcat访问url：<http://localhost:8080/generator/table/list>

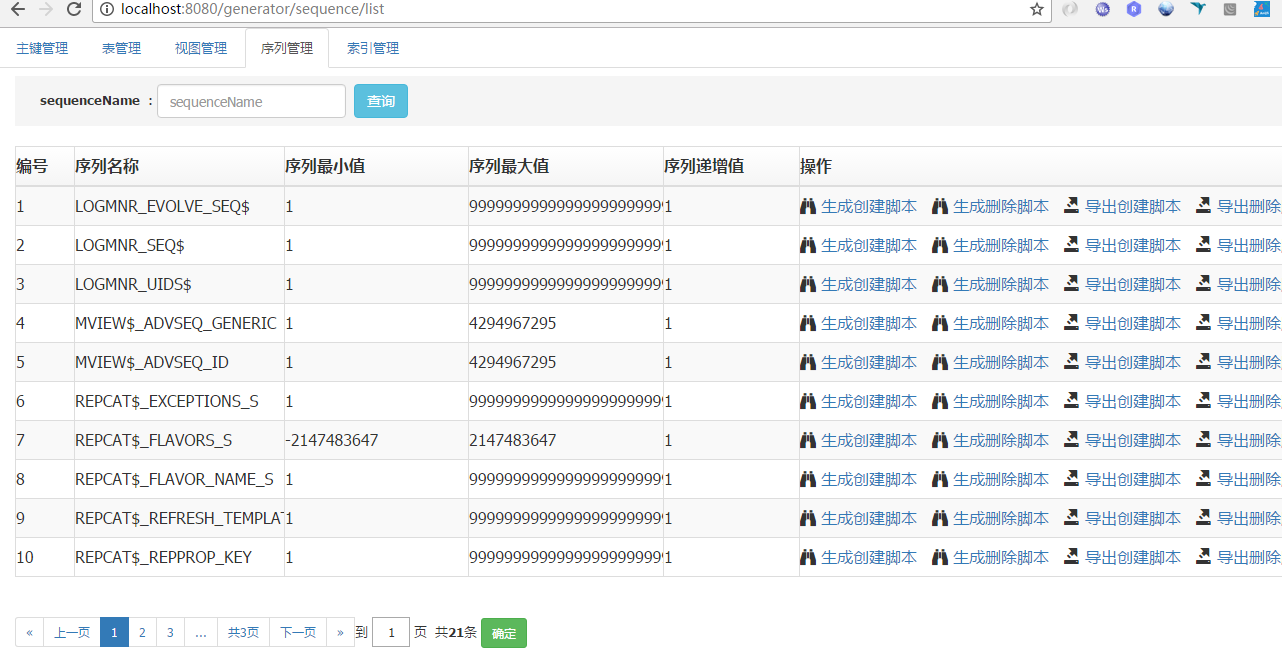
1. 整体功能--表管理



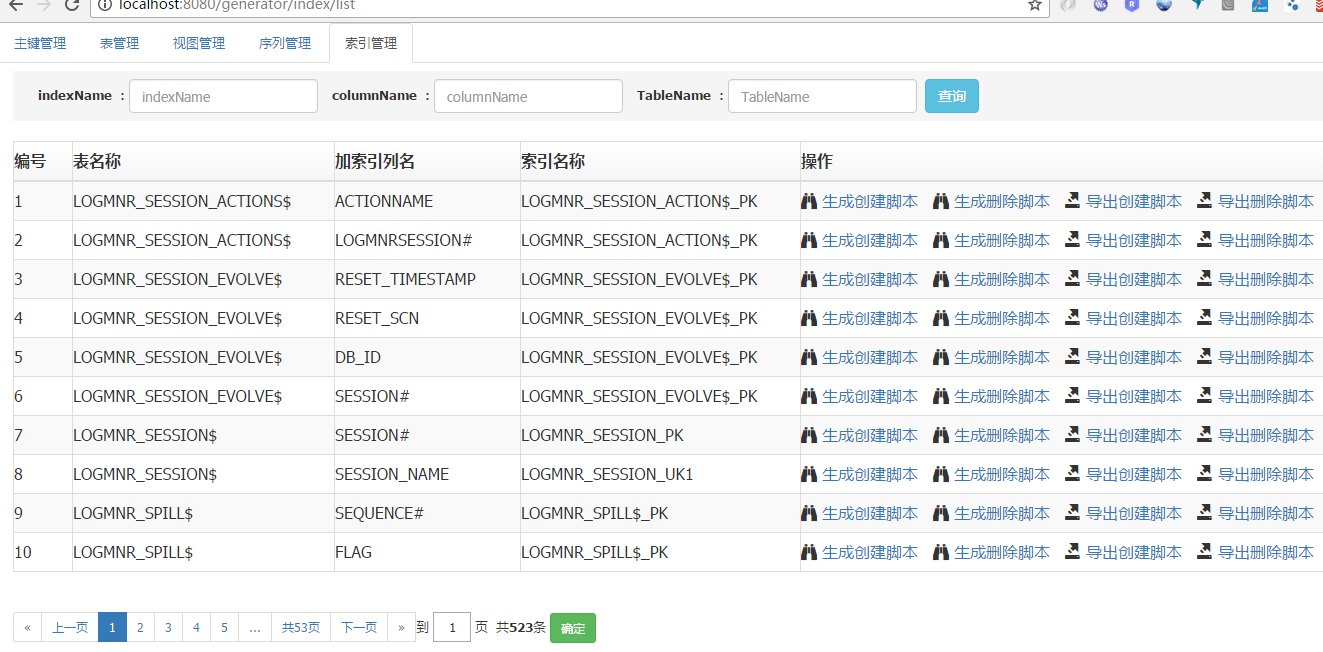
1. 整体功能--视图管理



1. 整体功能--序列管理



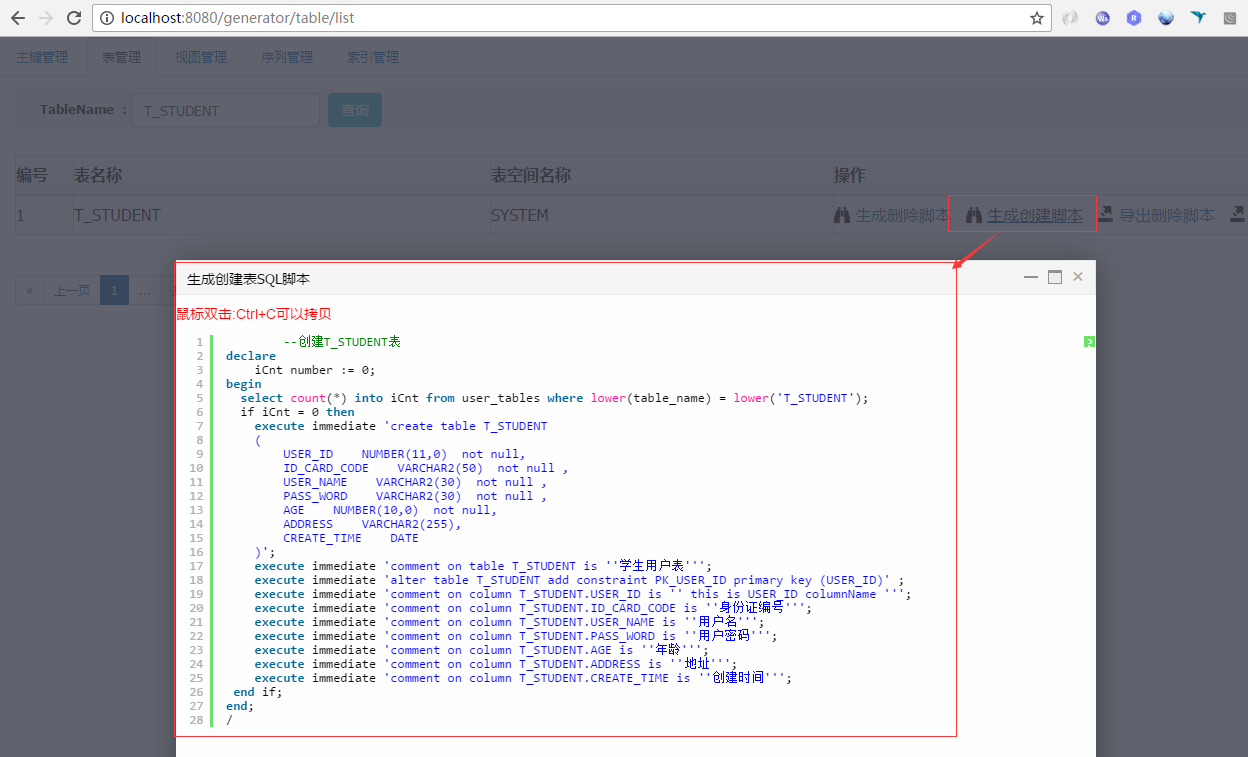
1. 整体功能--索引管理



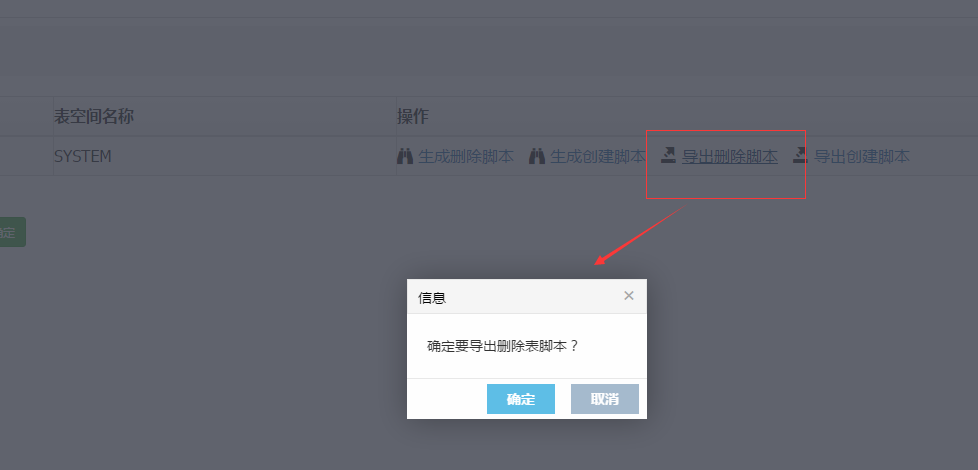
6、生成删除表SQL脚本

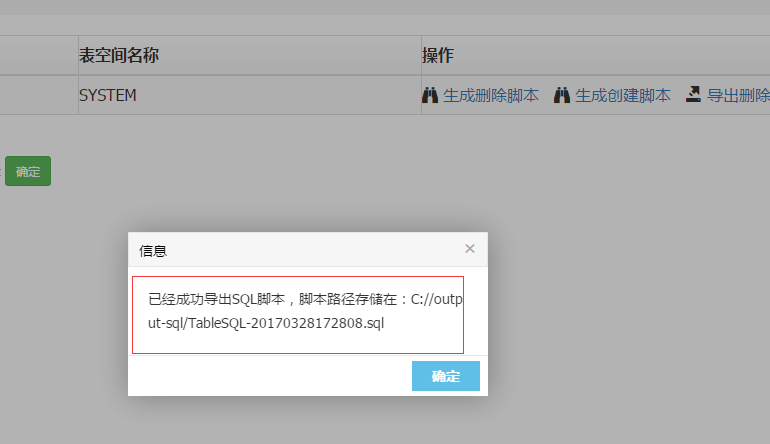


7、生成创建表SQL脚本

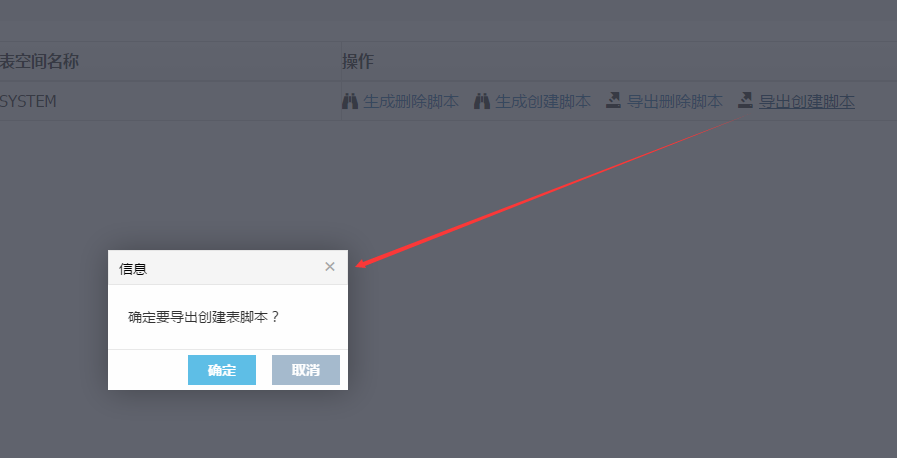


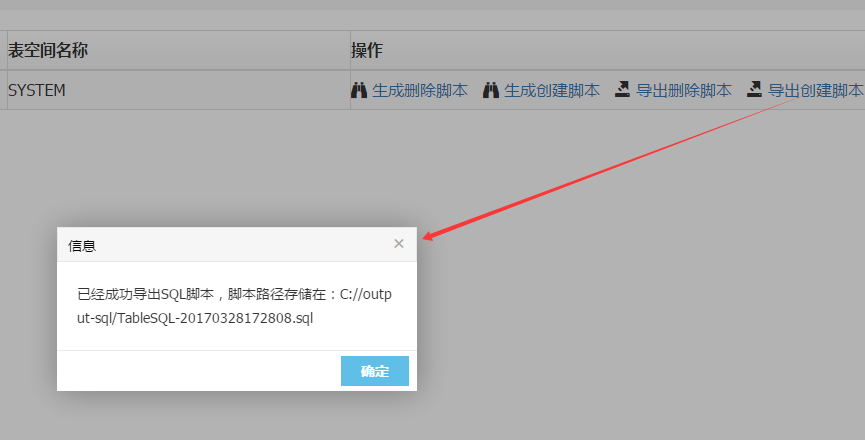
8、导出删除表SQL脚本

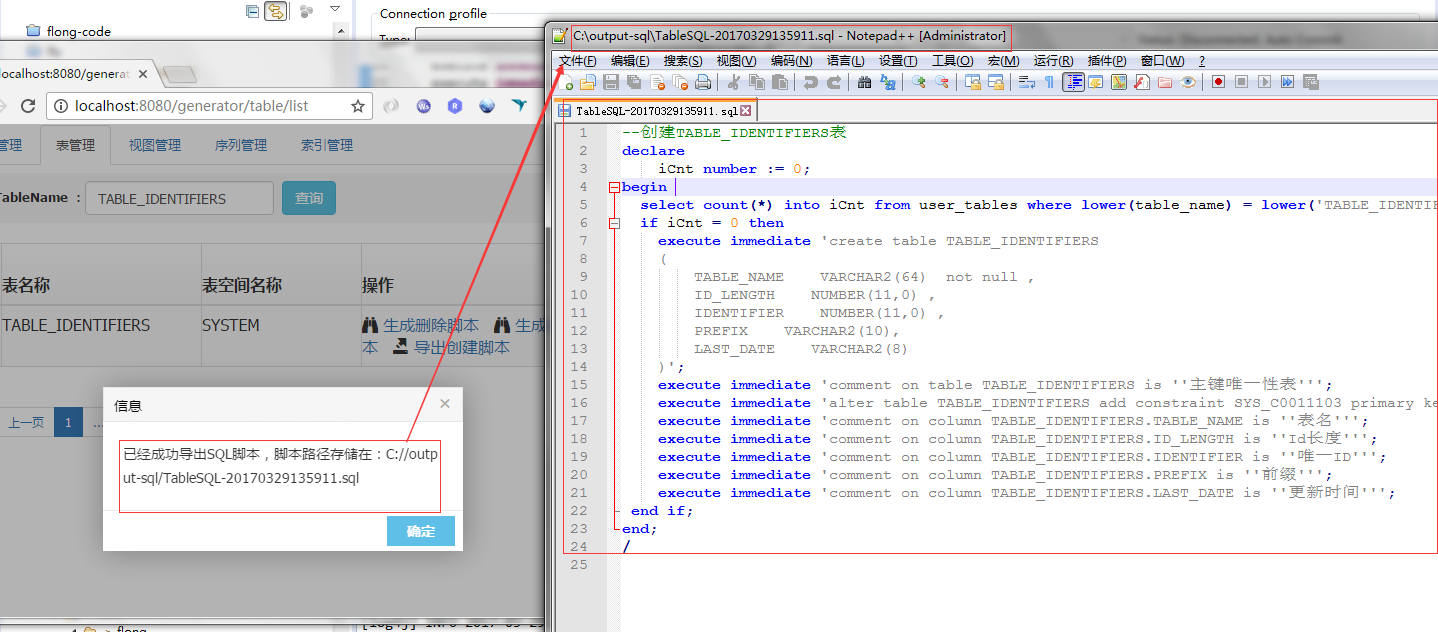




9、导出创建表SQL脚本







# 个人总结

## 7.1为什么要总结

1. 常言道：一分耕耘一分收获。总结能肯定自己劳动成果，帮助自己找出优缺点。
2. 在扬长避短的情况下不断完善自己不足，针对性的学习。
3. 个人认为：在学习过程中只有不断的总结与定位自己现况的位置，根据自身的情况督促自己进行选择性的学习来提升自己不足非常重要。

## 7.2为什么记录笔记

1. 在平时工作下来，一天认真工作下来，脑子肯定非常很疲倦，故经常健忘。也许跟年龄的岁数增大有关系。人精力与体力等各方面都有限的，毕竟人不是机器。
2. 常言道：好记性不如烂笔头。切记千万别以自己有点小聪明对自己懒惰。
3. 如果经常健忘的时候，不防借助一些软件工具或者云服务进行帮助存储生活中的痕迹。比如：思维导图（XMind, MindManager,FreeMind）云道笔记等等方式。

## 7.3其他说明

1以上问题都是根据自己工作实际情况进行总结整理，除了技术问题查很多网上资料通过进行学习之后梳理。

2在学习过程中也遇到很多困难和疑点，如有问题或误点，望各位老司机多多指出或者提出

建议。本人会采纳各种好建议和正确方式不断完善现况，人在成长过程中的需要优质的养料。

3 导入代码的时候遇到最多的问题，我想应该是Maven较多，此时不懂maven的童鞋们可以通过自身情况，进行网上查资料学习。如通过网上找资料长时间解决不了，或者框架有不明白可以通过[博客留言](http://jilongliang.iteye.com)或者jilongliang@sina.com邮箱进行提问,在能力范围内会尽力帮助大家解决问题所在，希望在过程中一起进步，一起成长。请牢记一句话：内事问度娘, 外事问谷歌，仅供参考学习。