**Jess杰思框架**

**开发手册**

**山东鲁能智能**

**2015年3月**

目录

[1. 目录结构 3](#_Toc413999114)

[1.1. JESS框架目录结构 3](#_Toc413999115)

[1.2. 源码目录 3](#_Toc413999116)

[1.3. 资源目录 4](#_Toc413999117)

[1.4. 发布目录 4](#_Toc413999118)

[1.5. 执行目录 4](#_Toc413999119)

[2. 环境搭建 5](#_Toc413999120)

[2.1. 准备工作 5](#_Toc413999121)

[2.2. 导入项目 5](#_Toc413999122)

[2.3. 导入数据表 5](#_Toc413999123)

[2.3.1. Mysql数据库 5](#_Toc413999124)

[2.3.2. Oracle数据库 7](#_Toc413999125)

[2.4. 部署项目 7](#_Toc413999126)

[3. 快速开发 8](#_Toc413999127)

[3.1. 单表场景 8](#_Toc413999128)

[3.2. 主从表场景 11](#_Toc413999129)

[3.3. 可编辑列表场景 14](#_Toc413999130)

[4. 代码生成器中级联关系解析 16](#_Toc413999131)

[5. 交流、反馈、参与贡献 17](#_Toc413999132)

# 目录结构

## JESS框架目录结构

|  |  |
| --- | --- |
| jess | 项目根目录 |
| ├ apiDoc | 标准doc API开发文档目录 |
| ├ db | 数据库初始化脚本目录 |
| ├ helpDoc | 业务开发文档等帮助文档目录 |
| ├ src | 源码目录 |
| └ target | 单元测试或者发布的资源目录 |

## 源码目录

**src/main/java**

|  |  |
| --- | --- |
| com.lnint.jess |  |
| ├ common | 公共模块存放目录 |
| │ ├ beanvalidator | 实体Bean验证相关类 |
| │ ├ config | 全局配置类 |
| │ ├ filter | 页面缓存过滤器 |
| │ ├ log | 日志工具相关类 |
| │ ├ mapper | 各种Object到Xml、Object到Json的映射转换类 |
| │ ├ persistence | 持久层相关类 |
| │ ├ security | 安全相关类 |
| │ ├ service | 业务层相关类 |
| │ ├ servlet | 公共servlet相关类 |
| │ ├ utils | 各种操作小工具类 |
| │ └ web | 模型控制器层相关类 |
| ├ core | Jess内置功能模块存放目录 |
| │ ├ sys | 系统管理模块存放目录 |
| │ │ ├ dao | 数据访问层相关类 |
| │ │ ├ entity | 实体相关类 |
| │ │ ├ interceptor | 系统模块拦截器相关类 |
| │ │ ├ service | 业务处理相关类 |
| │ │ ├ web | 模型控制器层相关类 |
| │ │ └ utils | 系统模块的工具类 |
| │ ├ generate | 代码生成器存放目录 |
| │ │ └ template | 代码生成器模板目录 |
| │ └ demo | 示例代码存放目录 |
| └ modules | 业务功能模块存放目录 |

## 资源目录

**src/main/resource**

|  |  |
| --- | --- |
| cache | Ehcache缓存配置存放目录 |
| META-INF | 资源相关信息路径 |
| application.properties | Spring配置属性文件 |
| applicationContext-shiro.xml | Spring配置安全配置文件 |
| applicationContext.xml | Spring主配置文件 |
| log4j.properties | Log4j日志配置属性文件 |

## 发布目录

**src/main/webapp**

|  |  |
| --- | --- |
| static | 静态文件存放目录 |
| └ compressor.bat | JavaScrpt和CSS文件压缩脚本 |
| userfiles | 用户上传文件目录 |
| WEB-INF | WEB应用安全目录，通过映射访问相关文件。 |
| ├ lib | 依赖jar包目录 |
| ├ tags | Tags标签存放目录 |
| ├ tlds | Tld文件存放目录 |
| ├ views | 视图文件目录 |
| │ ├ error | 系统异常映射相关页面 |
| │ ├ include | 视图相关包含文件 |
| │ ├ layouts | 视图布局相关文件 |
| │ ├ core | Jess内置功能模块视图相关文件 |
| │ ├ sys | 系统管理模块视图相关文件 |
| │ ├ generate | 代码生成器视图相关文件 |
| │ └ demo | 示例模块视图相关文件 |
| │ └ modules | 业务功能模块视图相关文件 |
| ├ ckfinder.xml | CKfinder配置文件 |
| ├ decorators.xml | Decorator配置文件 |
| ├ spring-mvc.xml | Spring MVC配置文件 |
| └ web.xml | Web配置文件 |

## 执行目录

**bin**

|  |  |
| --- | --- |
| refresh-db | 数据库相关执行文件 |
| ├ cms | 各模块数据初始化目录，包括：MySql、Oracle、MsSql建表脚本，数据初始数据文件。 |
| ├ oa |
| ├ sys |
| └ refresh-db.bat | 刷新数据库执行脚本（需要安装Maven） |
| eclipase.bat | 生成eclipse项目执行脚本 |
| install.bat | 生成编译项目 |
| jetty.bat | Jetty服务器运行脚本 |
| tomcat6.bat | Tomcat6服务器运行脚本 |
| tomcat7.bat | Tomcat7服务器运行脚本 |

# 环境搭建

## 准备工作

1. 安装JDK1.6，下载路径：http://jdk6.java.net/download.html
2. 安装和配置Maven3.0，下载路径：http://maven.apache.org/download.cgi
3. 安装好JDK和Maven后，打开pom.xml文件，修改“artifactId”节点改为你的项目名，例如：<artifactId>jess</artifactId>。
4. 运行bin\eclipse.bat脚本，等待执行完成即可（第一次运行，需要下载依赖jar包比较慢，请耐心等待）。

## 导入项目

1. 安装Eclipse，下载路径：http://www.eclipse.org/downloads/
2. 安装Tomcat6或Tomcat7最新版，

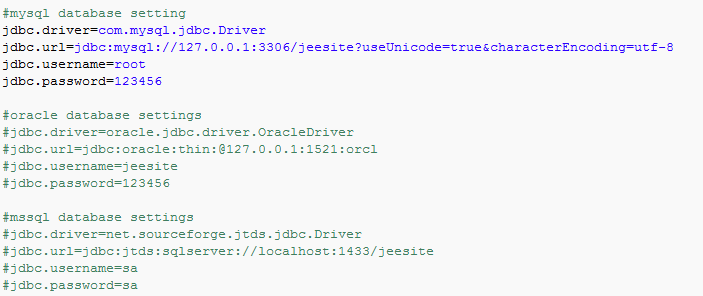
下载路径：http://tomcat.apache.org/download-60.cgi

1. 打开Eclipse集成开发环境，配置JDK和Tomcat，然后依次操作：File --> Import --> Existing Projects in Workspace，然后选择下载Jess项目目录，导入即可。

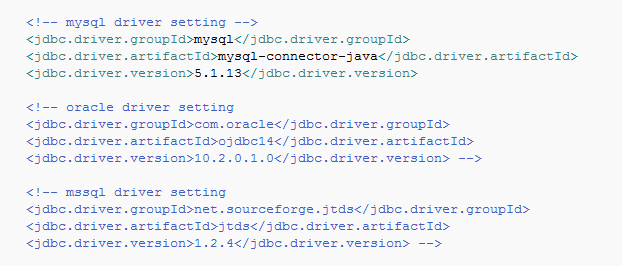
## 导入数据表

### Mysql数据库

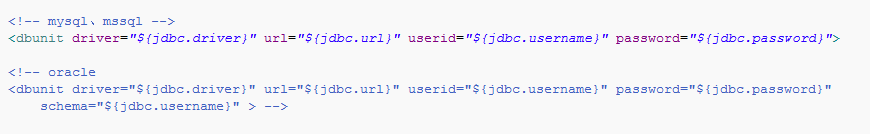
1. 修改src\main\resources\jess.properties文件中的数据库设置参数：



1. 修改pom.xml的数据库驱动：







1. 根据修改参数创建对应数据库（数据库编码：UTF-8）。
2. 执行数据库初始化脚本initdb\_mysql.sql，即可导入表结构及演示数据。

### Oracle数据库

1. 修改pom.xml文件

注释掉：

<!--<jdbc.driver.groupId>mysql</jdbc.driver.groupId>

<jdbc.driver.artifactId>mysql-connector-java</jdbc.driver.artifactId>

<jdbc.driver.version>5.1.13</jdbc.driver.version>-->

去掉注释：

<jdbc.driver.groupId>com.oracle</jdbc.driver.groupId>

<jdbc.driver.artifactId>ojdbc14</jdbc.driver.artifactId>

<jdbc.driver.version>10.2.0.1.0</jdbc.driver.version>

注释掉：

<!--<property name="sql.type" value="mysql" />

<property name="dbunit.datatype" value="org.dbunit.ext.mysql.MySqlDataTypeFactory" /> -->

去掉注释：

<property name="sql.type" value="oracle" />

<property name="dbunit.datatype" value="org.dbunit.ext.oracle.Oracle10DataTypeFactory" />

1. 修改src\main\resources\jess.properties文件中的数据库设置参数：

注释掉：

#jdbc.driver=com.mysql.jdbc.Driver

#jdbc.url=jdbc:mysql://127.0.0.1:3306/jess?useUnicode=true&characterEncoding=utf-8

#jdbc.username=root

#jdbc.password=123456

去掉注释：

#oracle database settings

jdbc.driver=oracle.jdbc.driver.OracleDriver

jdbc.url=jdbc:oracle:thin:@127.0.0.1:1521:orcl

jdbc.username=jess

jdbc.password=123456

1. 根据修改参数创建对应数据库
2. 执行数据库初始化脚本initdb\_oracle.sql，即可导入表结构及演示数据。

## 部署项目

1. 编译项目。
2. 将项目部署到Tomcat，修改Tomcat的Server.xml文件，Connector节点增加URIEncoding="UTF-8"属性，并启动服务
3. 打开浏览器输入发布地址，例如：http://127.0.0.1:8080/jess。

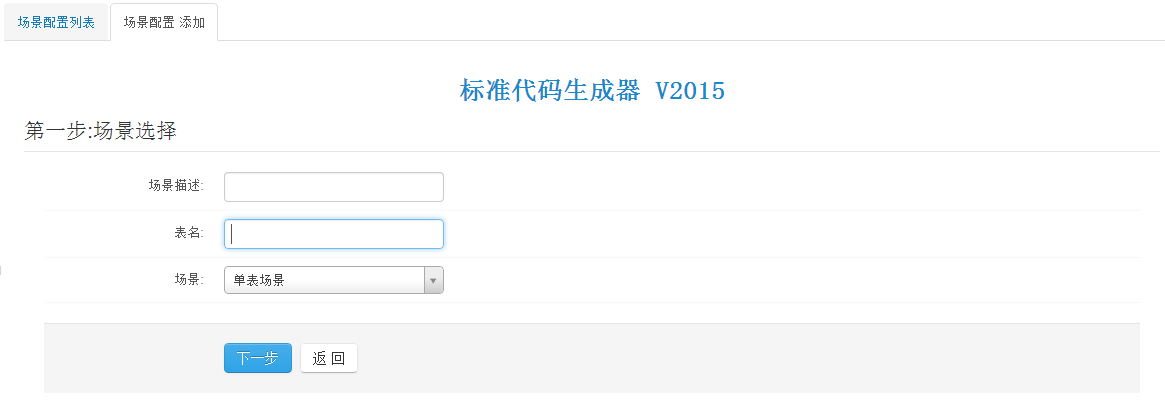
# 快速开发

## 单表场景

1. 点击“系统设置”—“代码生成”—“场景配置”，如下图所示：



1. 点击“场景配置添加”，进入第一步“场景选择”，如下图所示：



填写场景描述；

填写表名，如“gen\_scene” ；

选择场景，目前有单表、主从表、可编辑列表三种场景，选择“单表场景”；

1. 点击“下一步”，进入第二步“信息配置”，如下图：





“主表类名”，即生成的实体类名。

“页记录数”，即生成的list页面的每页的记录数，如10

列表中列出gen\_scene表中的字段，根据要生成的entity和jsp页面在列表中进行填写。

“字段名称”在jsp页面中显示的名称；

“JAVA字段”在entity和jsp页面中字段的名称；

“级联关系”有“无”、“一对一”、“多对一”三个选项；

“级联实体”如果“级联关系”选择“无”，则不可用；如果“级联关系”选择“一对一”和“多对一”，则填入级联的实体类名；

“级联操作”如果“级联关系”选择“无”，则不可用；如果“级联关系”选择“一对一”和“多对一”，有“无操作”、“级联删除”、“级联刷新”、“级联新增”、“级联修改”、“ALL”六个选项；

“Entity显示”如果勾选，则在entity实体类中显示；

“可为空”如果勾选，则在jsp页面中不必填；

“列表显示”如果勾选，则在list页面中显示；

“查询条件”和“模糊查询”如果勾选，则在list页面中作为查询条件并且模糊查询；

“表单显示”如果勾选，则在form表单中显示；

“显示控件”即在查询条件和form表单中显示的控件类型，有“文本框”、“文本域”、“下拉框”、“日期”四种选项；

“字典名”只有“显示控件”为“下拉框”时才可用，填入对应的数据字典名。

如果级联关系是“一对一双向关系被维护端”、“一对多单向”、“一对多双向”、“多对多”、“多对多关系被维护端”，点击“添加级联”，下边的列表中添加一行记录

“级联关系”有“一对一双向关系被维护端”、“一对多单向”、“一对多双向”、“多对多”、“多对多关系被维护端”五种选项；

“级联实体”为级联的实体类名；

“joinColumn”级联关系为“一对多单向”时需要维护，即关联字段的名称。级联关系为“多对多”时需要维护中间表中对应本实体的字段名；

“mappedBy”级联关系为“一对一双向关系被维护端”时需要维护，即在级联的实体类中对应本实体对象的对象名；

“joinTable”级联关系为“多对多”时需要维护，即中间关系维护表的表名；

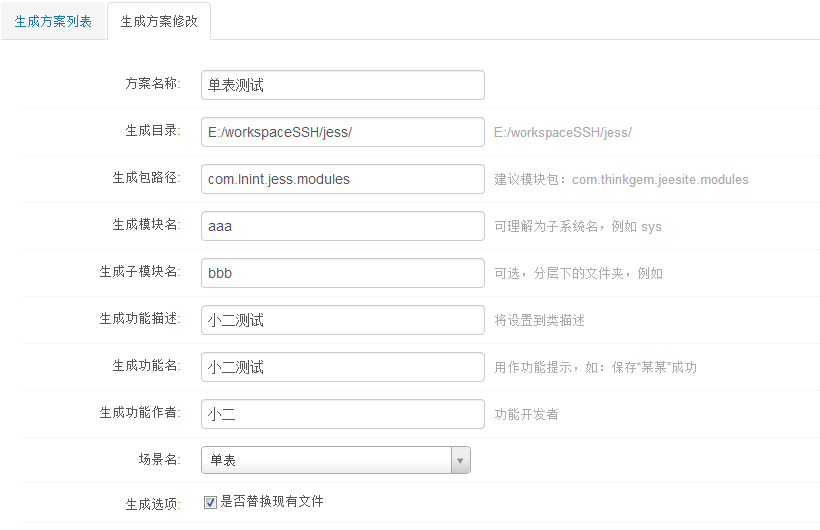
“inversejoinColumn” 级联关系为“多对多”时需要维护中间表中对应级联实体的字段名；

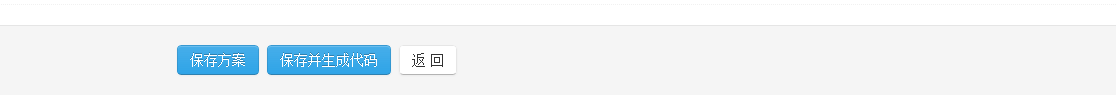
“级联操作”同上。

1. 点击“系统设置”—“代码生成”—“生成方案配置”，如下图所示：



点击“生成方案添加”，如下图所示：





1. 点击“保存并生成代码”，代码生成器会为你自动建立以下文件：

* 实体类：

jess\src\main\java\com\lnint\jess\modules\aaa\entity\bbb\MyTest.java

* 数据访问类：

jess\src\main\java\com\lnint\jess\modules\aaa\dao\bbb\MyTestDao.java

* 业务逻辑类：

jess\src\main\java\com\lnint\jess\modules\aaa\service\bbb\MyTestService.java

* 视图控制器类：

jess\src\main\java\com\lnint\jess\modules\aaa\web\bbb\MyTestController.java

* 视图文件（表单）：

jess\src\main\webapp\WEB-INF\views\modules\aaa\bbb\myTestForm.jsp

* 视图文件（列表）：

jess\src\main\webapp\WEB-INF\views\modules\aaa\bbb\myTestList.jsp

com\lnint\jess\modules是第四步中的包名；

aaa是第四步中的模块名；

bbb是第四步中的子模块名；

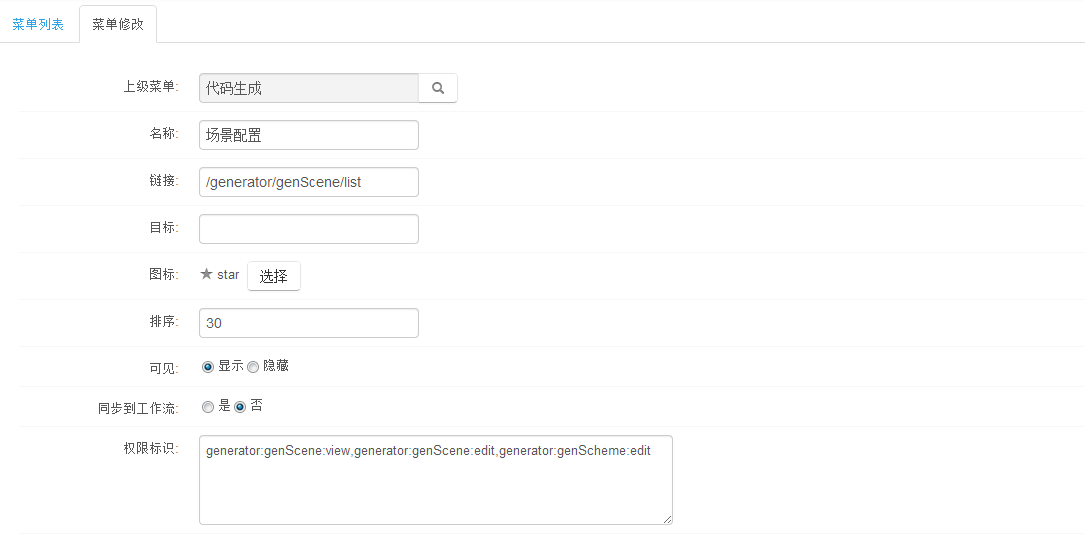
MyTest是第三步中的类名。

1. 配置菜单

点击“系统设置”—“系统设置”—“菜单管理”



点击“菜单添加”，如下图所示：



“名称” 菜单的名称

“链接” 生成的controller类里@RequestMapping里${adminPath}后边的部分再加上“/list”，eg：/generator/genScene/list

“可见”选择显示

“权限标识”生成的controller类里所有的@RequiresPermissions，用逗号分隔，eg：generator:genScene:view,generator:genScene:edit

## 主从表场景

1. 点击“系统设置”—“代码生成”—“场景配置”，如下图所示：



1. 点击“场景配置添加”，进入第一步“场景选择”，如下图所示：



填写表名，如“gen\_scene”

选择场景，选择“主从表场景”。

填写从表名，如“gen\_scene\_column”

1. 点击“下一步”，进入第二步“信息配置”，如下图：





分别填写“主表实体模型信息”和“从表实体模型信息”两个tab选项卡，具体内容参照单表场景中的信息配置。

“主表类名”，即生成的主表实体类名。

“页记录数”，即生成的list页面的每页的记录数，如10

作为主表，需要与从表维护一个“一对多关系被维护端”的级联关系，如上图所示，添加级联，级联关系选择“一对多关系被维护端”，级联实体填写从表实体类的名称，eg：ChildTest，mappedBy填写在从表实体类里对应本实体对象的对象名，eg：fatherTest。

其余参照“单表场景”





“从表类名”，即生成的从表实体类名。

“页记录数”，即生成的formlist页面列表每页的记录数，如10

作为从表，需要与主表维护一个“多对一”的级联关系，如上图所示，选择“多对一”，级联实体填写主表实体类的名称，eg：FatherTest。

1. 点击“系统设置”—“代码生成”—“生成方案配置”，如下图所示：



点击“生成方案添加”，同“单表场景”

1. 点击“保存并生成代码”，代码生成器会为你自动建立以下文件：

* 实体类：

jess\src\main\java\com\lnint\jess\modules\aaa\entity\bbb\FatherTest.java

jess\src\main\java\com\lnint\jess\modules\aaa\entity\bbb\ChildTest.java

* 数据访问类：

jess\src\main\java\com\lnint\jess\modules\aaa\dao\bbb\FatherTestDao.java

jess\src\main\java\com\lnint\jess\modules\aaa\dao\bbb\ChildTestDao.java

* 业务逻辑类：

jess\src\main\java\com\lnint\jess\modules\aaa\service\bbb\FatherTestService.java

jess\src\main\java\com\lnint\jess\modules\aaa\service\bbb\ChildTestService.java

* 视图控制器类：

jess\src\main\java\com\lnint\jess\modules\aaa\web\bbb\FatherTestController.java

jess\src\main\java\com\lnint\jess\modules\aaa\web\bbb\ChildTestController.java

* 视图文件（从表表单）：

jess\src\main\webapp\WEB-INF\views\modules\aaa\bbb\childTestForm.jsp

* 视图文件（主表列表）：

jess\src\main\webapp\WEB-INF\views\modules\aaa\bbb\fatherTestList.jsp

* 视图文件（主表表单）：

jess\src\main\webapp\WEB-INF\views\modules\aaa\bbb\fatherTestFormList.jsp

com\lnint\jess\modules是第四步中的包名；

aaa是第四步中的模块名；

bbb是第四步中的子模块名；

FatherTest是第三步中的类名。

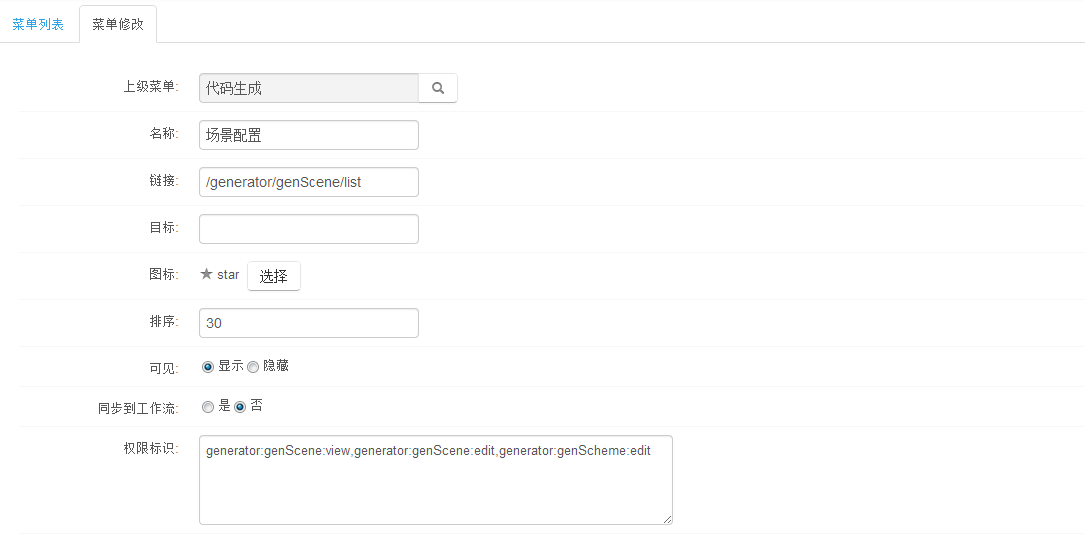
ChildTest是第三步中的子类名。

1. 配置菜单

点击“系统设置”—“系统设置”—“菜单管理”



点击“菜单添加”，如下图所示：



“名称” 菜单的名称

“链接” 生成的主表的controller类里@RequestMapping里${adminPath}后边的部分再加上“/list”，eg：/generator/genScene/list

“可见”选择显示

“权限标识”生成的主表和从表的controller类里所有的@RequiresPermissions，用逗号分隔，eg：generator:genScene:view,generator:genScene:edit

## 可编辑列表场景

1. 点击“系统设置”—“代码生成”—“场景配置”，如下图所示：



1. 点击“场景配置添加”，进入第一步“场景选择”，如下图所示：



填写场景描述；

填写表名，如“gen\_scene” ；

选择场景，目前有单表、主从表、可编辑列表三种场景，选择“可编辑列表”；

1. 点击“下一步”，进入第二步“信息配置”，如下图：



“主表类名”，即生成的实体类名。

“页记录数”，即生成的list页面的每页的记录数，如10

具体配置参考单表场景，区别是增加了“是否可修改”选项，去掉了“表单显示”选项。

“是否可修改”勾选，则在可编辑列表中可以进行编辑。

1. 点击“系统设置”—“代码生成”—“生成方案配置”，如下图所示：



点击“生成方案添加”，同“单表场景”

1. 点击“保存并生成代码”，代码生成器会为你自动建立以下文件：

* 实体类：

jess\src\main\java\com\lnint\jess\modules\aaa\entity\bbb\EditTest.java

* 数据访问类：

jess\src\main\java\com\lnint\jess\modules\aaa\dao\bbb\EditTestDao.java

* 业务逻辑类：

jess\src\main\java\com\lnint\jess\modules\aaa\service\bbb\EditTestService.java

* 视图控制器类：

jess\src\main\java\com\lnint\jess\modules\aaa\web\bbb\EditTestController.java

* 视图文件（可编辑列表）：

jess\src\main\webapp\WEB-INF\views\modules\aaa\bbb\editTestList.jsp

com\lnint\jess\modules是第四步中的包名；

aaa是第四步中的模块名；

bbb是第四步中的子模块名；

EditTest是第三步中的类名。

1. 配置菜单

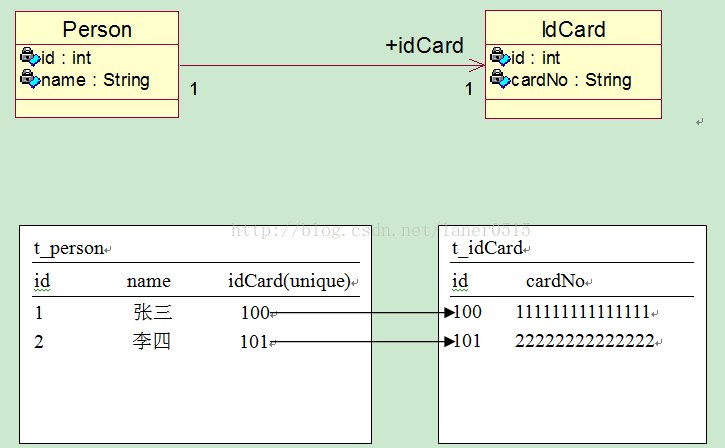
点击“系统设置”—“系统设置”—“菜单管理”

同“单表场景”

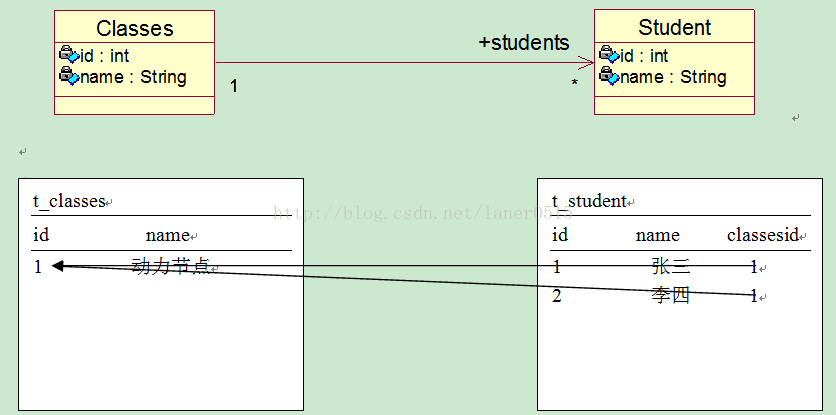
# 代码生成器中级联关系解析

## 表关联关系

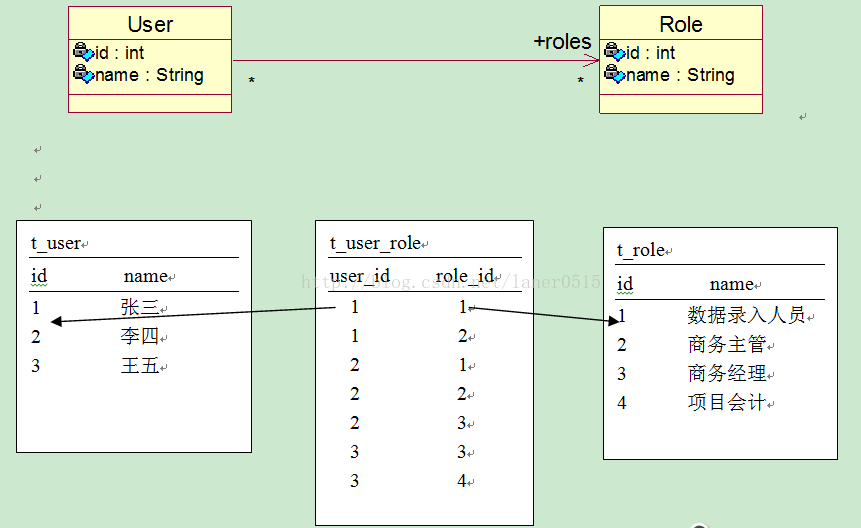
一对一



一对多



多对多



## Hibernate关系映射

1. 一对一单向外键关联

关系维护端

eg:

@OneToOne

@JoinColumn(name="wifeId")

对应我们级联关系中的“一对一”

由于是关系维护端,在列表中会列出外键字段,只需要在此字段上维护“级联关系”选择“一对一单向”；“级联实体”为Wife；“级联操作”为“ALL”.  
2. 一对一双向外键关联

关系维护端

eg:

@OneToOne

@JoinColumn(name="wifeId")

对应我们级联关系中“一对一”

关系被维护端

eg:

@OneToOne(mappedBy = "wife")

对应我们级联关系中“一对一双向关系被维护端”  
3. 一对多单向关联

关系被维护端

eg:

@OneToMany

@JoinColumn(name="departmentId")

对应我们级联关系中的“一对多单向”  
4. 多对一单向关联

关系维护端

eg:

@ManyToOne

@JoinColumn(name = "departmentId")

对应我们级联关系中的“多对一”

5. 一对多或者多对一双向关联

关系被维护端

eg:

@OneToMany(mappedBy = "department",cascade={CascadeType.ALL})

对应我们级联关系中的“一对多双向”

关系维护端

eg:

@ManyToOne

@JoinColumn(name = "departmentId")

对应我们级联关系中的“多对一”  
6. 多对多单向关联

关系维护端

eg:

@ManyToMany

@JoinTable(name="teacher\_student",

joinColumns={@JoinColumn(name="teacherId")},

inverseJoinColumns={@JoinColumn(name="studentId")}

)

对应我们级联关系中的“多对多”  
7. 多对多双向关联.

关系维护端

eg:

@ManyToMany

@JoinTable(name="teacher\_student",

joinColumns={@JoinColumn(name="teacherId")},

inverseJoinColumns={@JoinColumn(name="studentId")}

)

对应我们级联关系中的“多对多”

关系被维护端

eg:

@ManyToMany(mappedBy="students")

对应我们级联关系中的“多对多双向关系被维护端”

对于 Hibernate 来说，常见的映射关系就是上面的7种关系，但对于数据库来说，其实就只有一种关系--外键关联. 另外 Hibernate 一对一还有主键关联方式，但几乎没被用过，所以略去。

# 交流、反馈、参与贡献

QQ群：

E-mail：

支持Jess：

如果你想参与进来共同完善它或有更好的建议，请联系我吧(^\_^)。