# 序言

TIMO后台管理系统，基于SpringBoot2.0 + Spring Data Jpa + Thymeleaf + Shiro 开发的后台管理系统，采用分模块的方式便于开发和维护，支持前后台模块分别部署，目前支持的功能有：权限管理、部门管理、字典管理、日志记录、文件上传、代码生成等，为快速开发后台系统而生的脚手架！

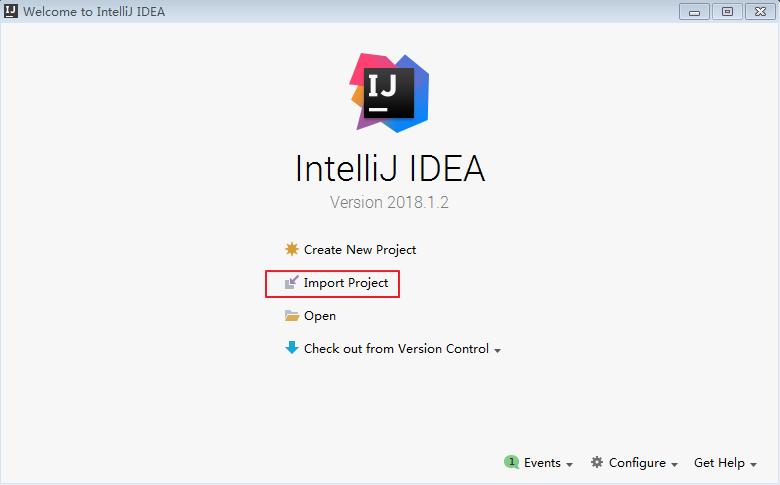
开发文档：[TIMO开发文档.看云](https://www.kancloud.cn/timo/timo-doc)

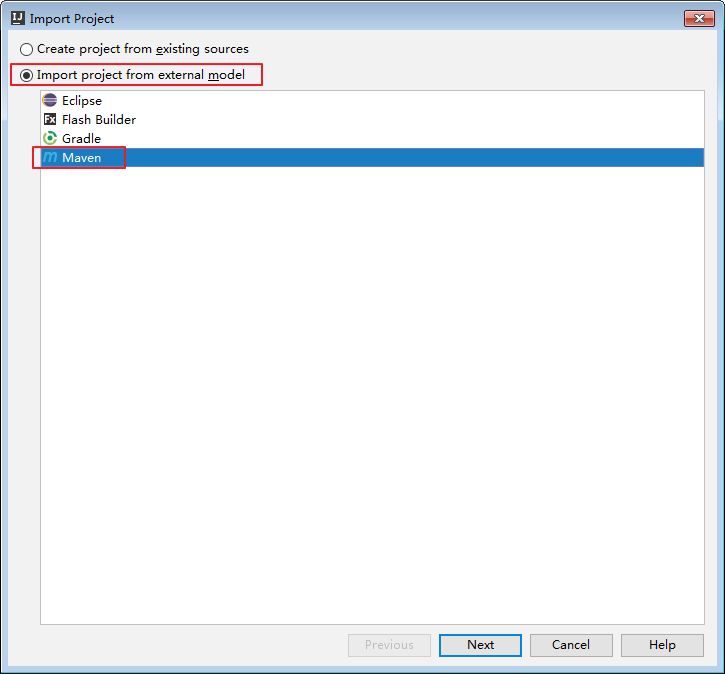
开源地址：<https://gitee.com/aun/Timo>

# 运行项目

## IntelliJ IDEA

### 导入项目





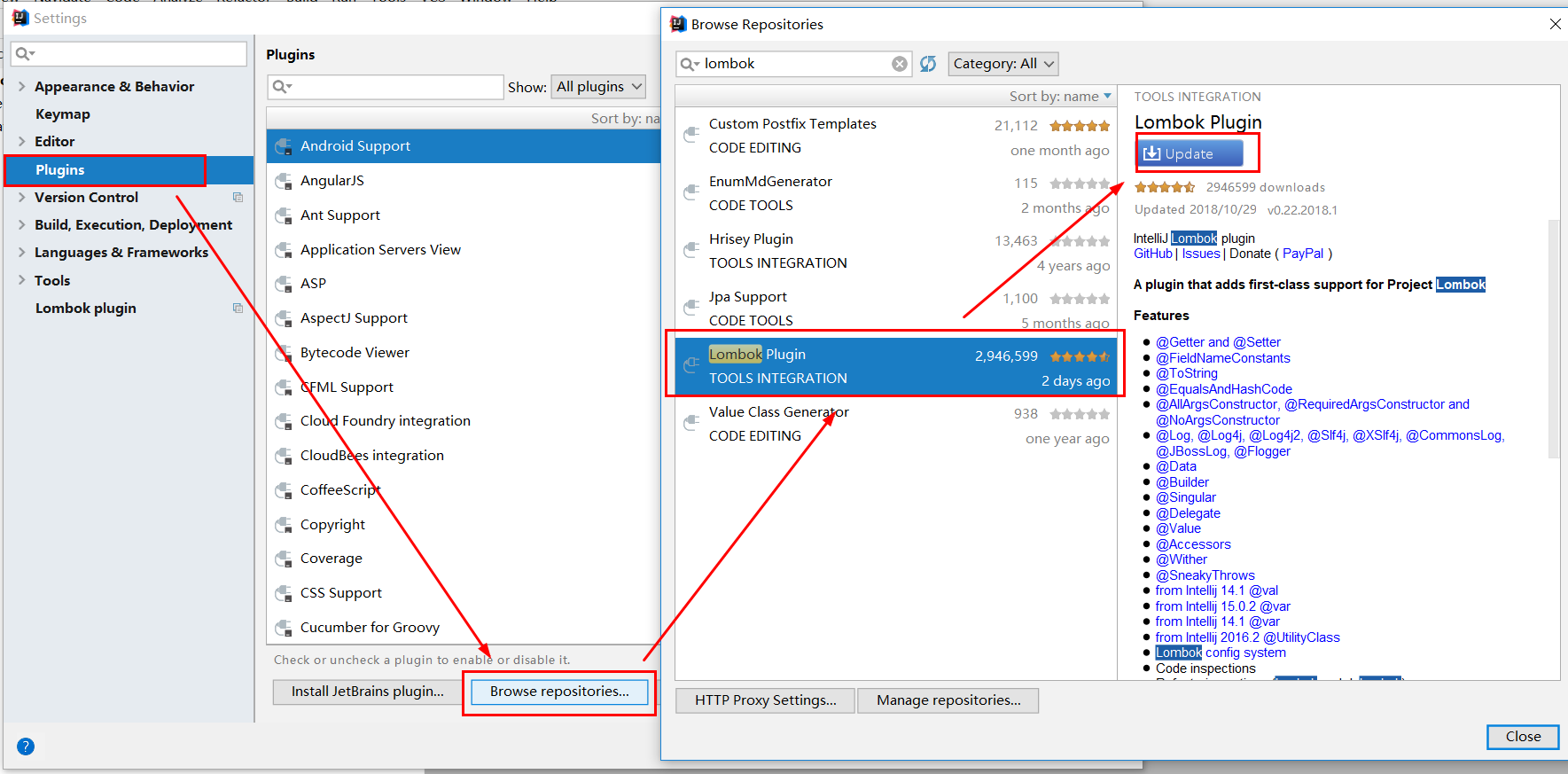
选择maven项目，一直按下一步即可

完成选择后需要等待一段时间，maven需要从网上下载相应的jar包

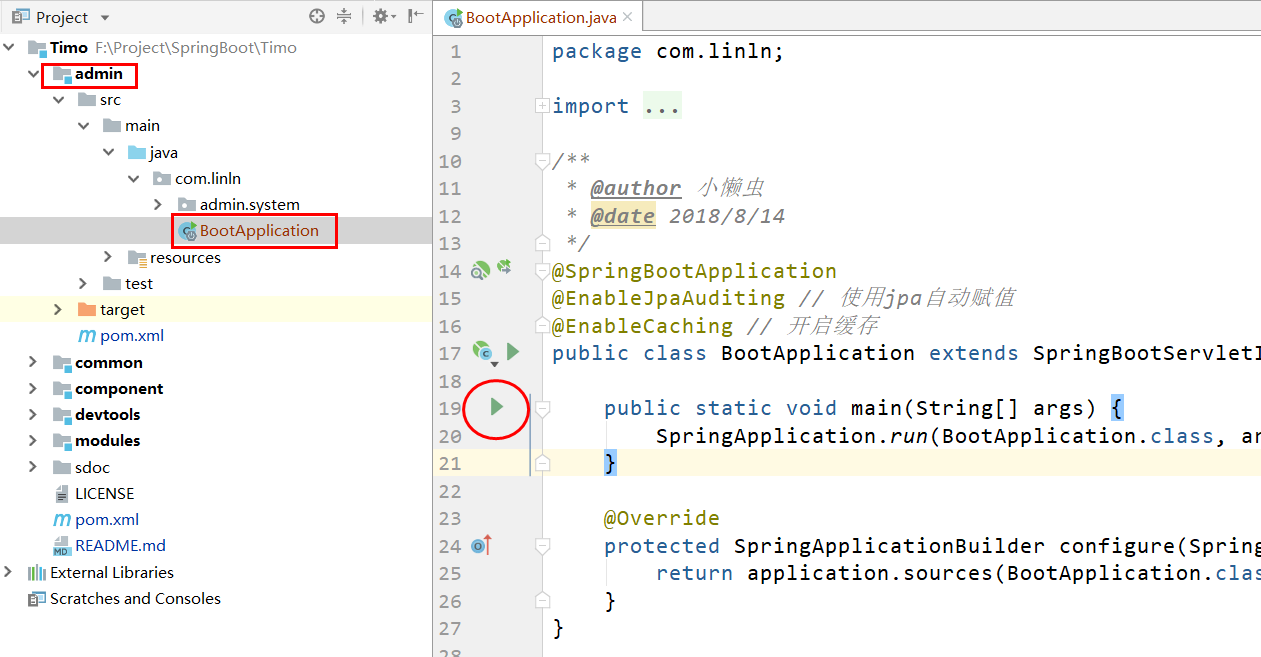
### 安装lombok插件

项目中使用了lombok，需要安装相应的插件支持

点击设置：file->Settings



### 运行项目-项目主入口



**默认数据库**：数据库名称timo 账号root 密码root

**访问路径**：<http://localhost:8080>

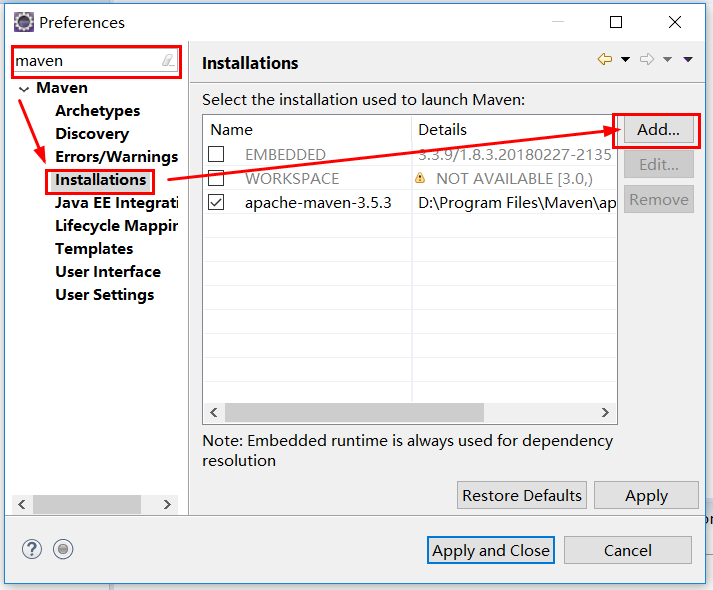
## Eclipse

### 安装Maven

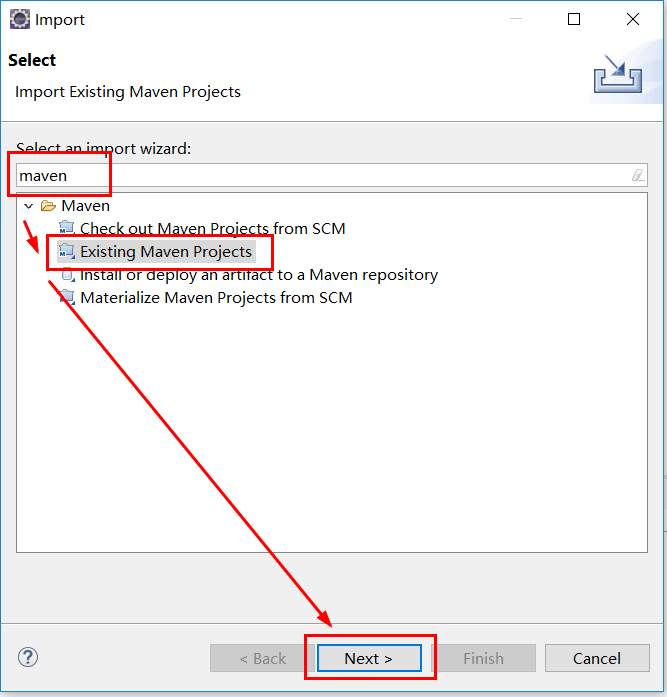
Eclipse自带maven工具，建议自行安装！

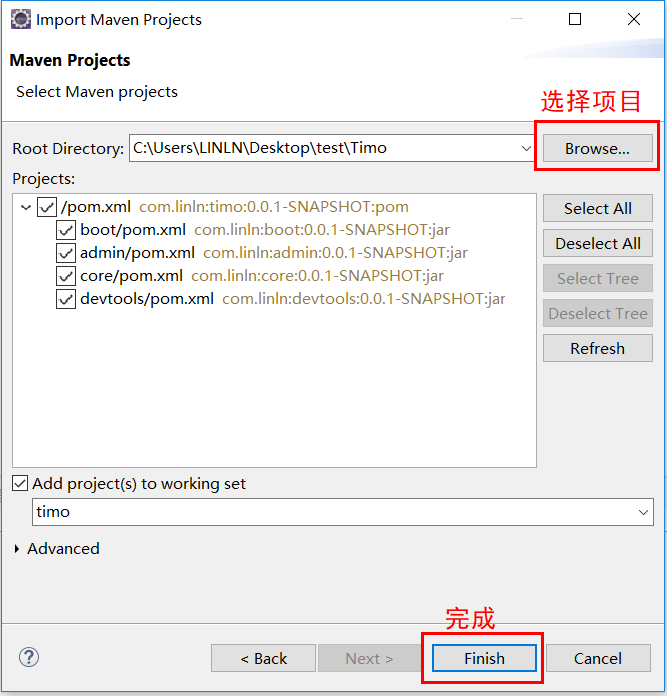
下载地址：<http://mirrors.tuna.tsinghua.edu.cn/apache/maven/maven-3/3.5.4/binaries/apache-maven-3.5.4-bin.zip>

下载完成解压即可！



### 导入项目





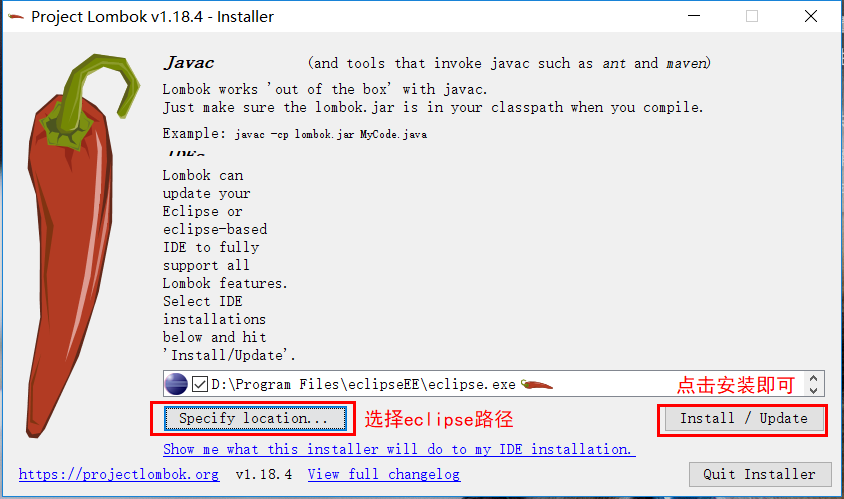
点击完成后需要等待一段时间， maven需要下载相应的依赖包

### 安装lombok插件

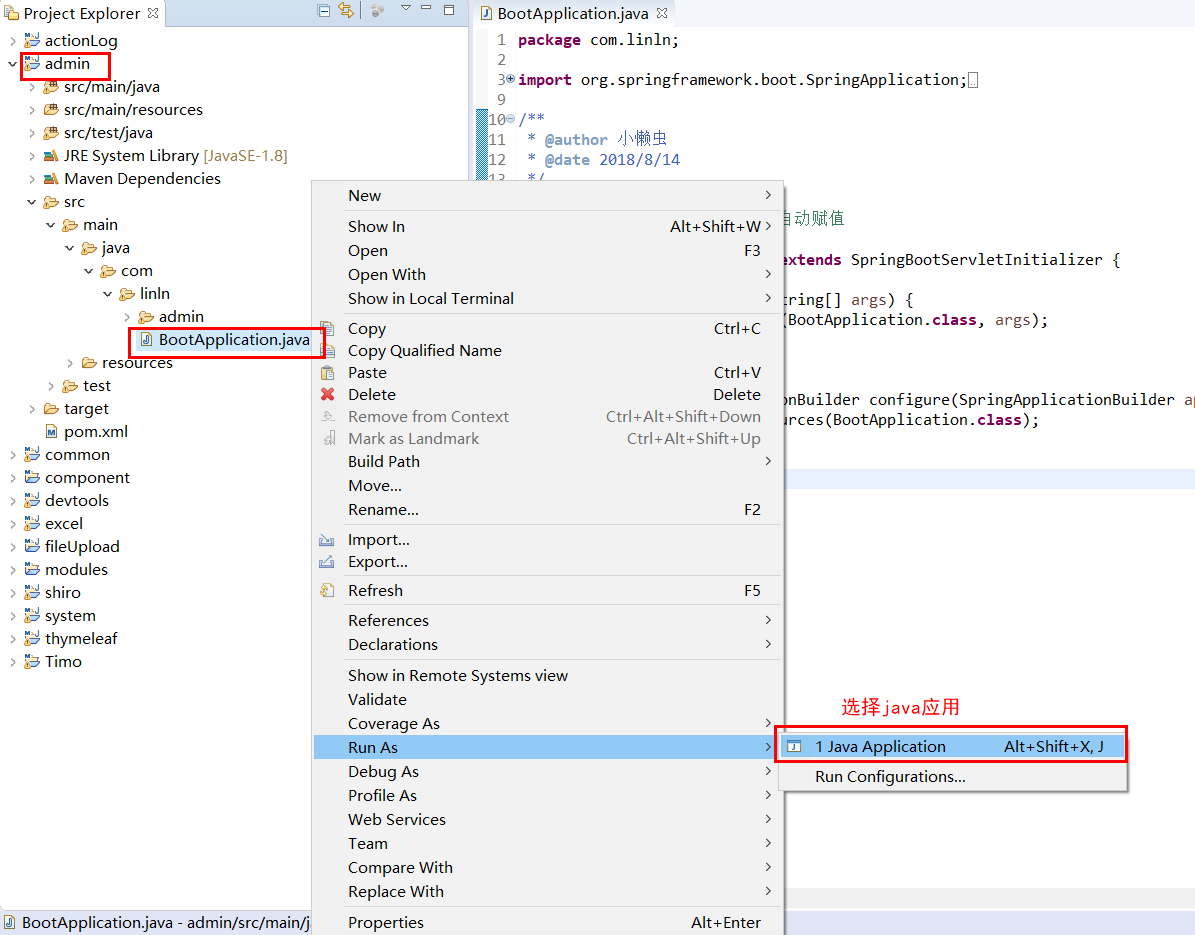
项目中使用了lombok，需要安装相应的插件支持

插件下载地址：<https://projectlombok.org/downloads/lombok.jar>

下载完成后点击运行，如果无法直接运行，请使用命令行java -jar lombok路径



### 运行项目-项目主入口



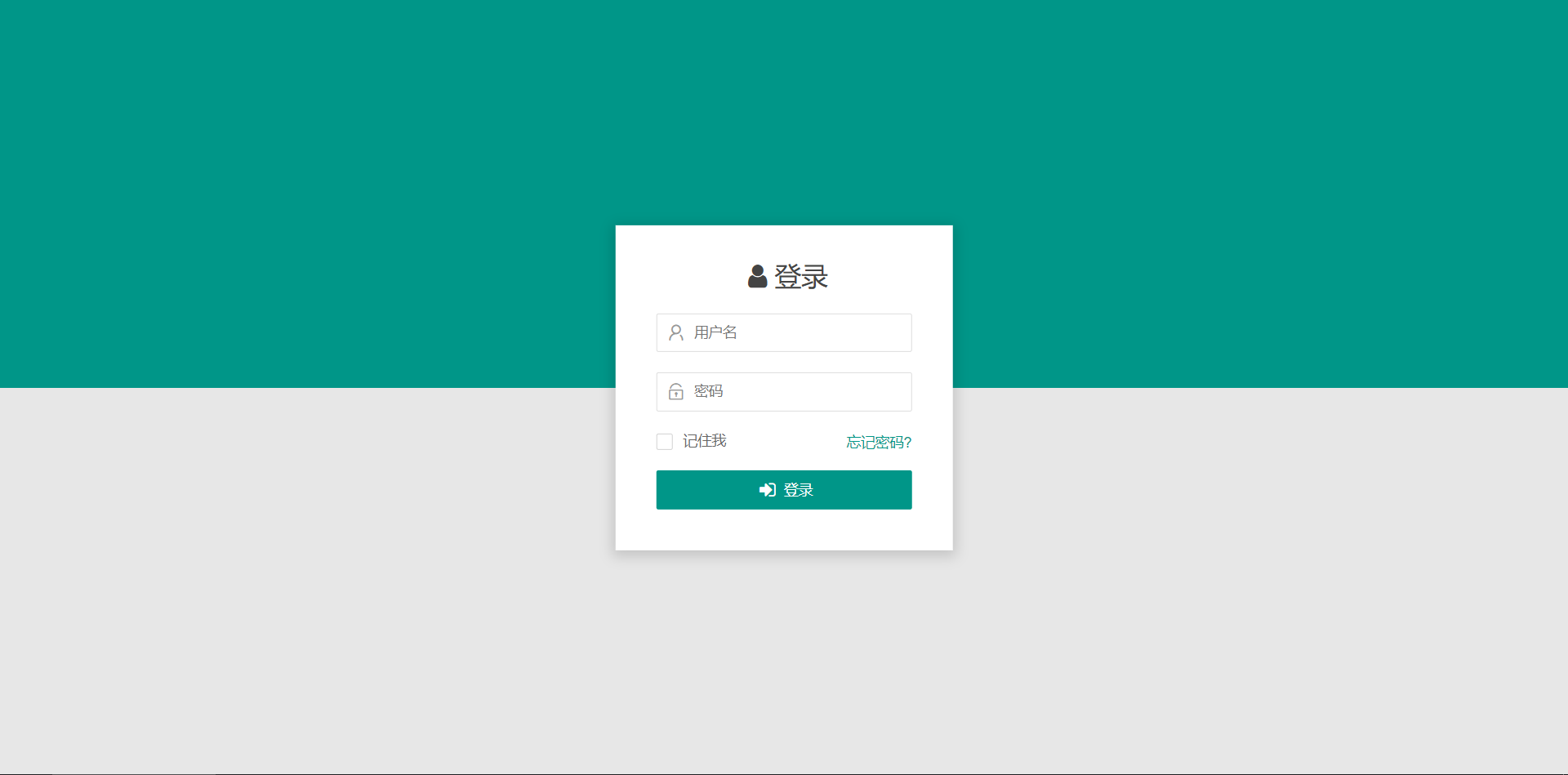
**默认数据库**：数据库名称timo 账号root 密码root

**访问路径**：<http://localhost:8080>

# 使用项目

## 登录入口

<http://localhost:8080/login>



账号：admin

密码：123456

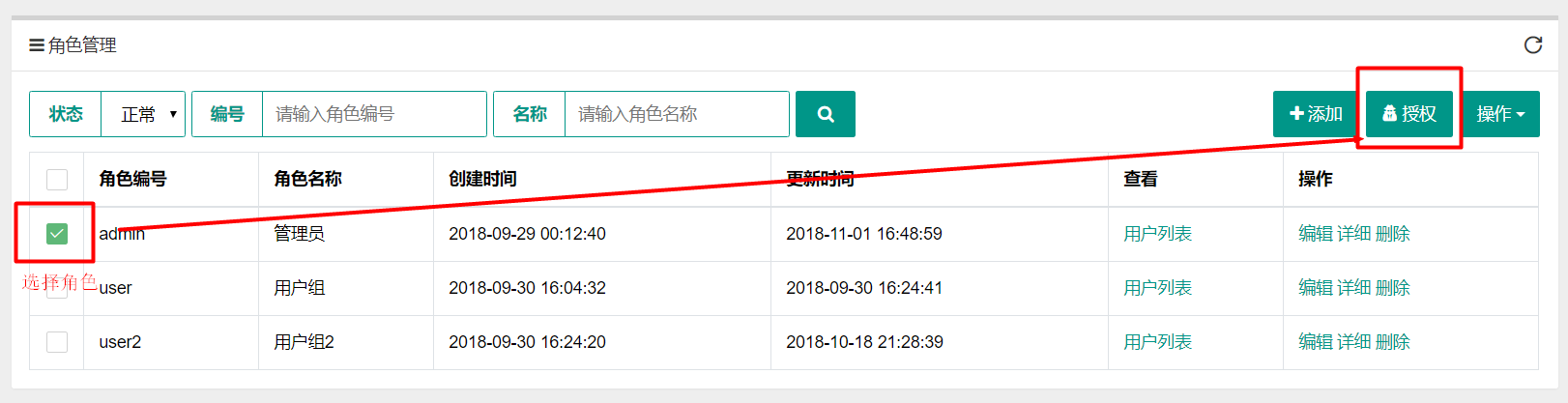
## 权限管理

本项目权限管理由3个模块组成，分别是：用户管理、角色管理、菜单管理（权限资源），同时权限管理也绑定了侧边栏菜单，如果权限发生改变需要重新登录才可正常显示。

1. 权限资源

本项目中的权限资源使用了菜单的地址，新增和删除权限资源请到菜单管理页面操作。

1. 权限资源分配



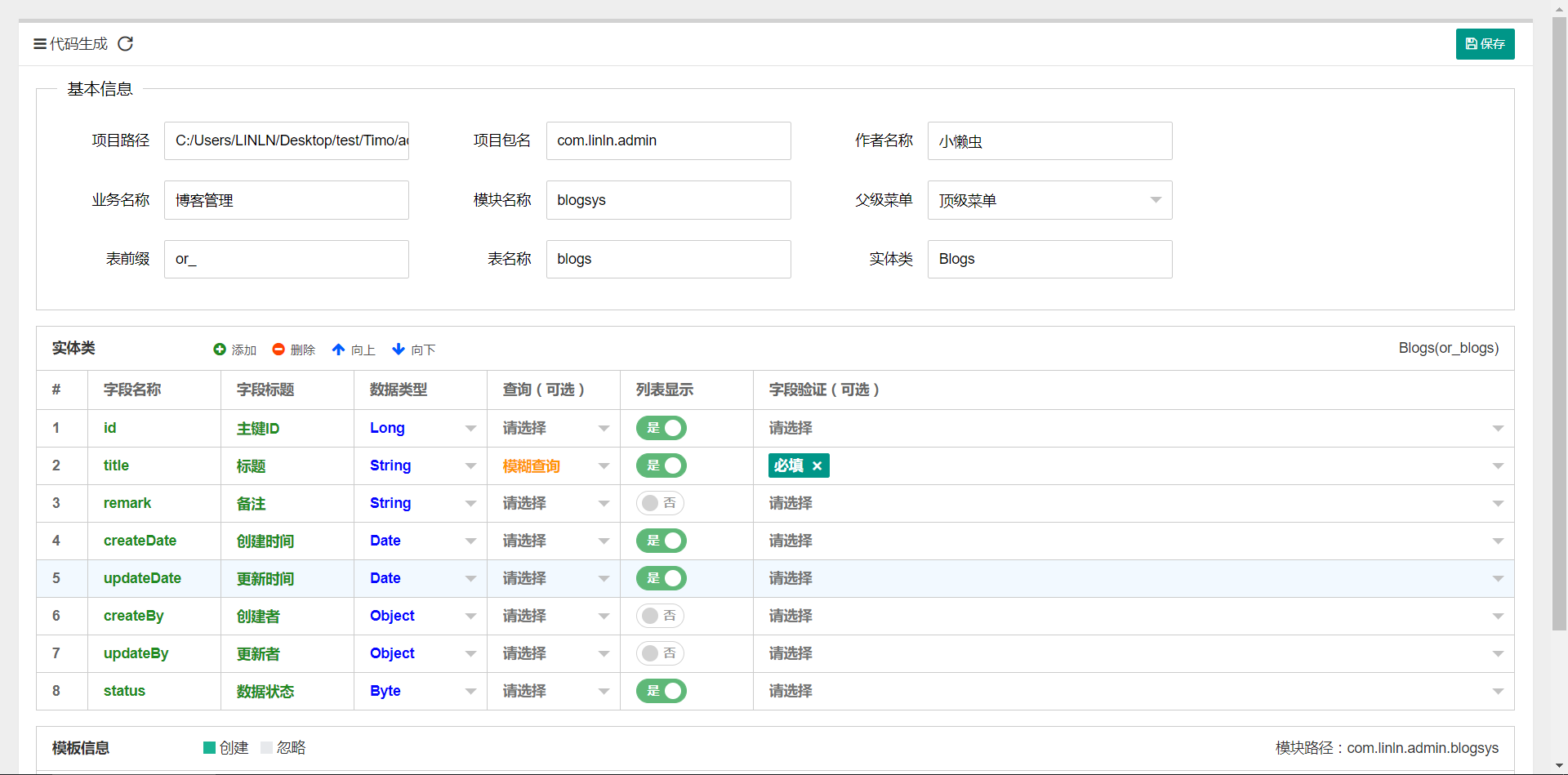


1. 角色分配





## 代码字段生成



### 基本信息

**项目路径**：项目所在的目录，一般没有特殊需要不需要修改

**项目包名**：项目包名，如果没有修改包路径，一般不需要修改

**作者名称**：生成文件用于备注

**父级菜单**：啊！！！！！！！！！！

**模块名称**：一个业务模块，如：博客系统、微信商城等

**业务名称**：如：用户管理、商品管理、商品分类等

**表前缀**：表前缀，用于数据库中区分不同的模块

**表名称**：不同单词间请使用“\_”连接

**实体类**：会根据表名称自动赋值，一般不需要修改

**访问地址**：用于控制基础的访问路径

**模块结构**：1.独立模块—会在业务组(modules)中生成一个新的模块(业务模块)，可以给前台模块使用！

2.后台模块—将全部文件生成到后台模块(admin)中，复用性不强，如果只是开发后台管理项目，可直接使用这个结构！

### 实体类

没啥好说的，建议保留id、remark、createDate、updateDate、createBy、createBy、status字段

### 模板信息

可以自行选择需要的模板，一般不需要去选择

### 注意！

代码生成后需要重新登录！重置权限和侧边栏菜单。Eclipse 需要刷新项目目录

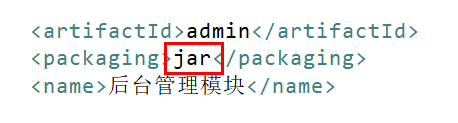
# 打包部署

## 说明

1. 支持两种打包方式：jar包、war包
2. 代码生成模块是一个独立的模块，在项目部署后一般不再使用，打包时可以将其依赖去掉，减少项目部署后的体积。

## Jar包方式

1. 进入admin模块下修改pom.xml文件，将其打包方式改为jar（<packaging>jar</packaging>）



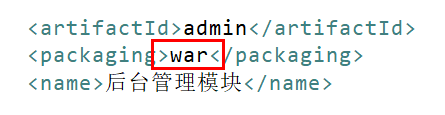
1. 使用maven命令行打包：mvn clean package

也可以通过IDE编辑器内置的maven打包方式

1. 在boot模块的target目录下找到打包后的jar包，通过java命令行运行即可

## War包方式

1. 进入boot模块下修改pom.xml文件，将其打包方式改为jar（<packaging>war</packaging>）



1. 使用maven命令行打包：mvn clean package

也可以通过IDE编辑器内置的maven打包方式

1. 在boot模块的target目录下找到打包后的war包，上传到服务器项目路径下即可。