构建spring-boot项目

Jdk环境

Jdk = [java工具](https://blog.csdn.net/u014042066/article/details/78831435)（编译工具） + jre + jvm

jvm作用：

而Java class文件是可以运行在任何支持Java[虚拟机](https://baike.baidu.com/item/%E8%99%9A%E6%8B%9F%E6%9C%BA" \t "https://baike.baidu.com/item/class%E6%96%87%E4%BB%B6/_blank)的硬件平台和操作系统上的二进制文件

Class文件需要经过jvm的解释才能被操作系统执行，jvm是操作系统和class文件的桥梁

jre是基础文件类库，jdk需要调用jre的lib文件才能解释Class文件，上层也能调用jre的文件

**反射**

JAVA反射机制是在运行状态中，对于任意一个类，都能够知道这个类的所有属性和方法。

这种动态获取的信息以及动态调用对象的方法的功能称为java语言的反射机制。

**Spring实现IOC(**控制反转**)**

控制反转，利用DI(依赖注入)，而DI由反射机制实现，由Ioc容器动态创建对象。

Spring 提供了 IoC容器通过配置文件或者注解的方式来管理对象之间的依赖关系。

传统创建实例：

public classUserService{

private UserDao userdao;

public void insert(User user){

userdao = new UserDao(); // 每次用都需要自己new一个UserDao类型的对象

userdao.insert(user);

}

}

IOC动态创建实例

public classUserService{

private UserDao userdao; // import进来的UserDao

public void insert(User user){

userdao.insert(user); // 不需要自己new对象了

}

}

**Spring Bean**

Bean有两种，IOC容器管理这些bean

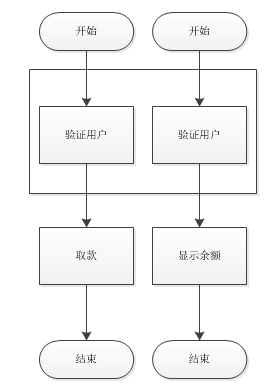
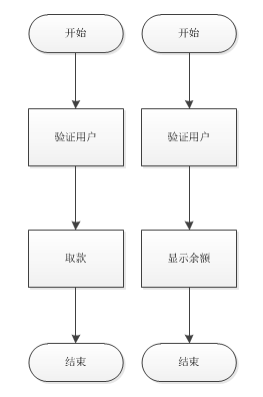
单例：模型提供了具有特定名称的对象的共享实例。

原型：模型确保每次检索都会创建单独的对象。

**Spring AOP 面向切面编程**

所以 AOP的本质是在一系列纵向的控制流程中，把那些相同的子流程提取成一个横向的面，面向这些横向的面进行操作。

比如 验证用户+逻辑



验证用户（公共代码）切面

对横切关注点或横切典型的职责分界线的行为（例如日志和事务管理）进行模块化

实现方式：

原理：动态代理机制+反射

动态代理机制：

AOP容器会动态创建一个代理对象供使用者调用，该代理对象已经按Java EE程序员的意图将切面成功切入到目标方法的连接点上，从而使切面的功能与业务逻辑的功能同时得以执行。

从原理上讲，调用者直接调用的其实是AOP容器动态生成的代理对象，再由代理对象调用目标对象完成原始的业务逻辑处理，而代理对象则已经将切面与业务逻辑方法进行了合成。