Day01笔记

知识点：

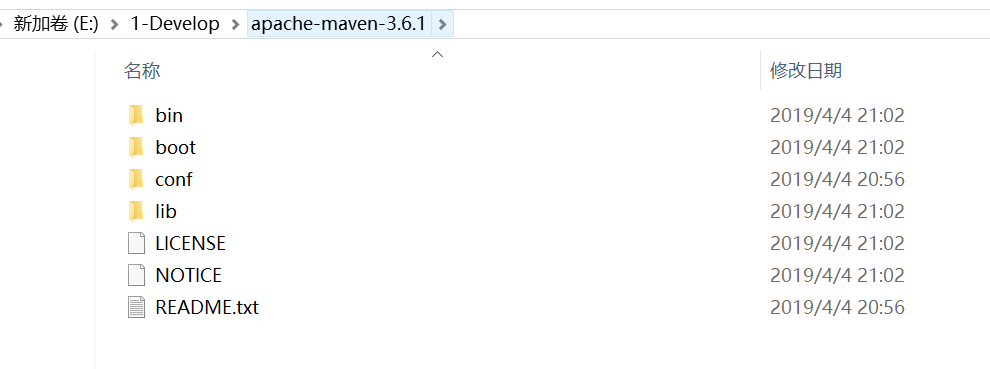
1. Maven的基础配置
2. 创建SpringBoot工程
3. Spring依赖注入知识点和Demo

**什么是Maven**

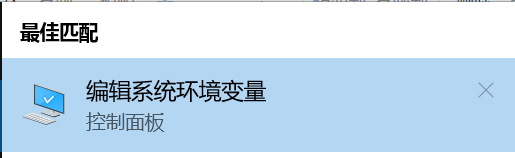
**[Maven](https://baike.baidu.com/item/Maven" \t "https://baike.baidu.com/item/Maven/_blank)项目对象模型(POM)，可以通过一小段描述信息来管理项目的构建，报告和文档的[项目管理工具](https://baike.baidu.com/item/%E9%A1%B9%E7%9B%AE%E7%AE%A1%E7%90%86%E5%B7%A5%E5%85%B7/6854630" \t "https://baike.baidu.com/item/Maven/_blank)软件**

Maven配置

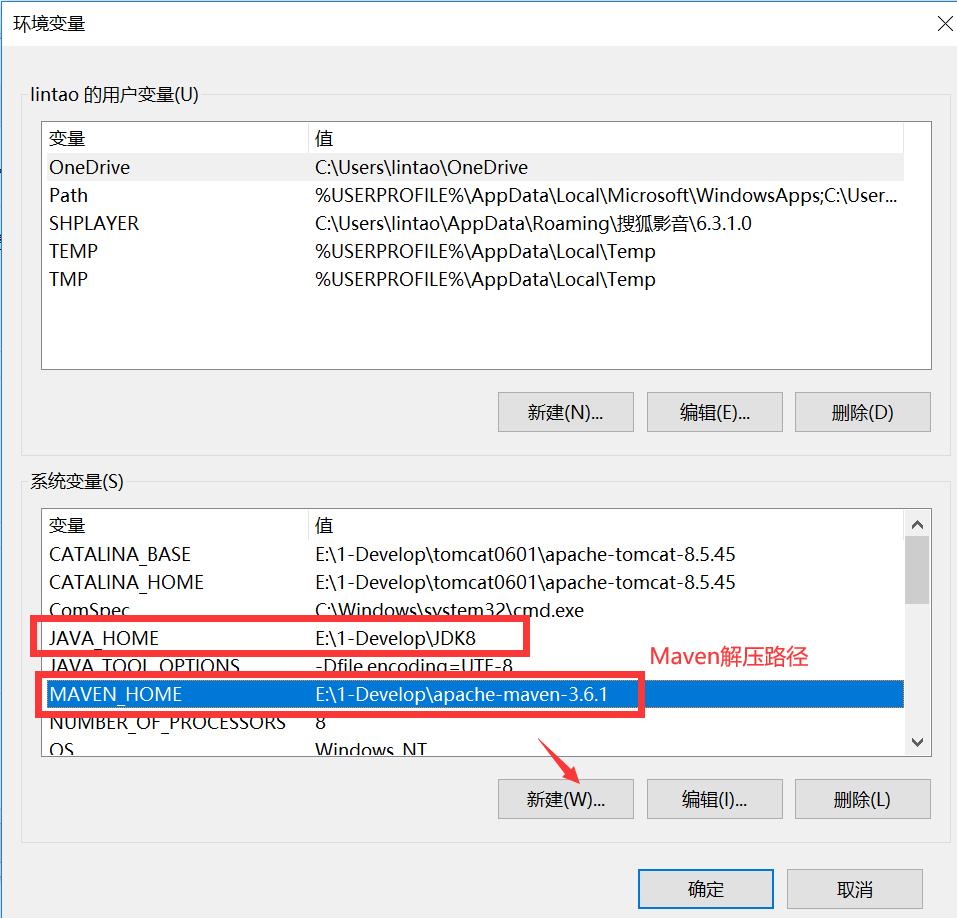
1. **解压maven**

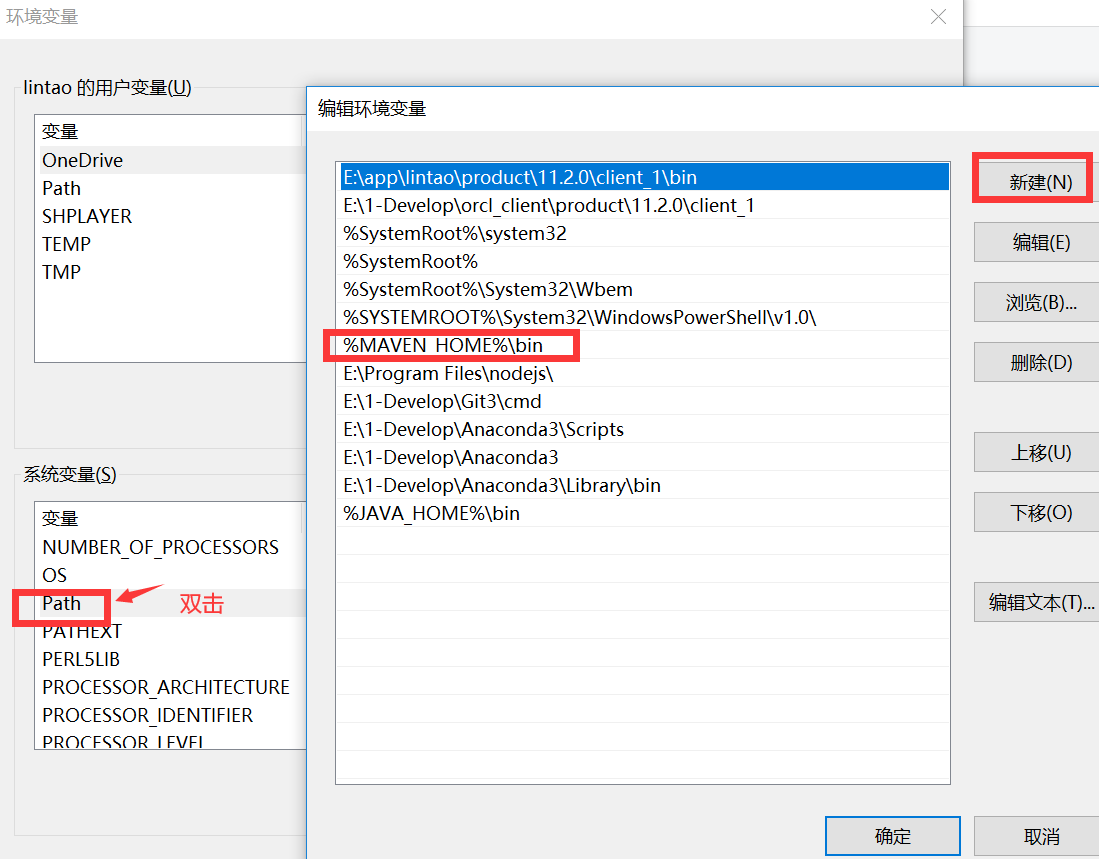


1. 配置环境变量



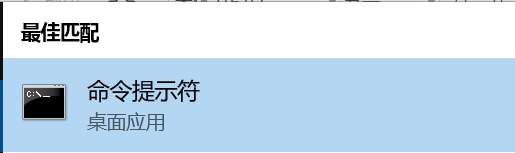






后面就是一路 “确定”

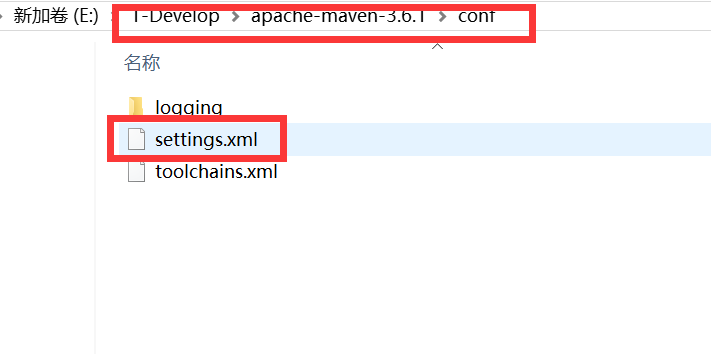
查看maven是否安装配置成功

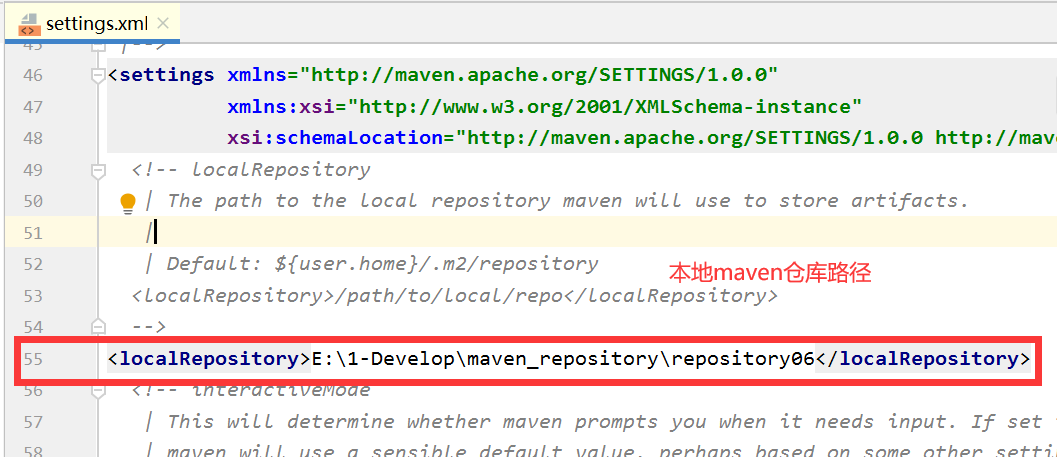




Maven已经配置好了

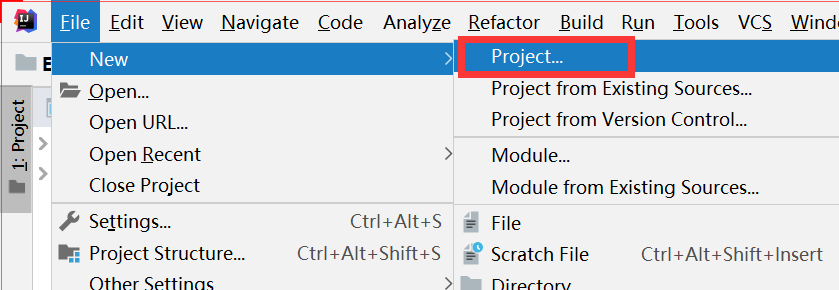
修改配置，设置本地maven仓库路径以及 远程仓库镜像设置为阿里云

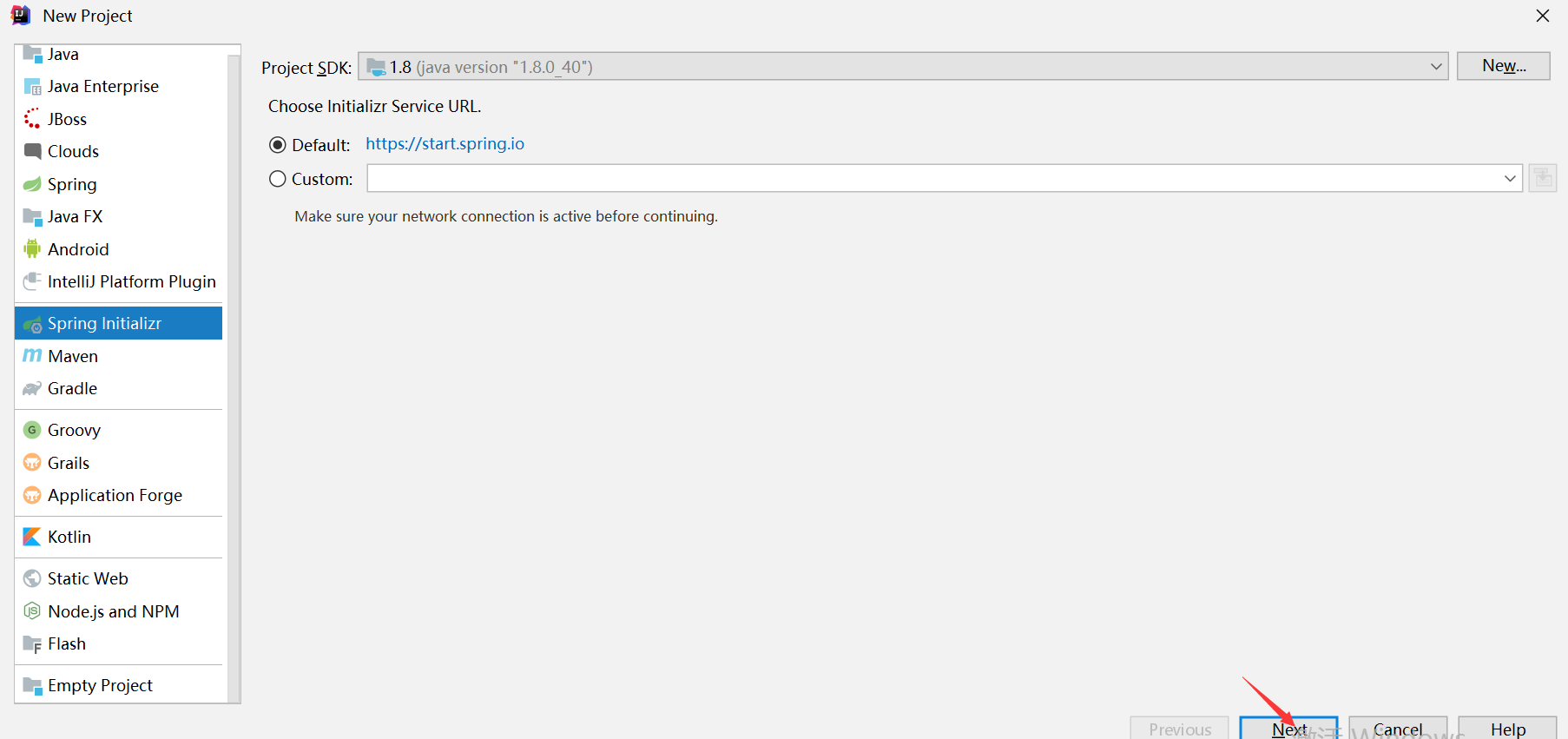


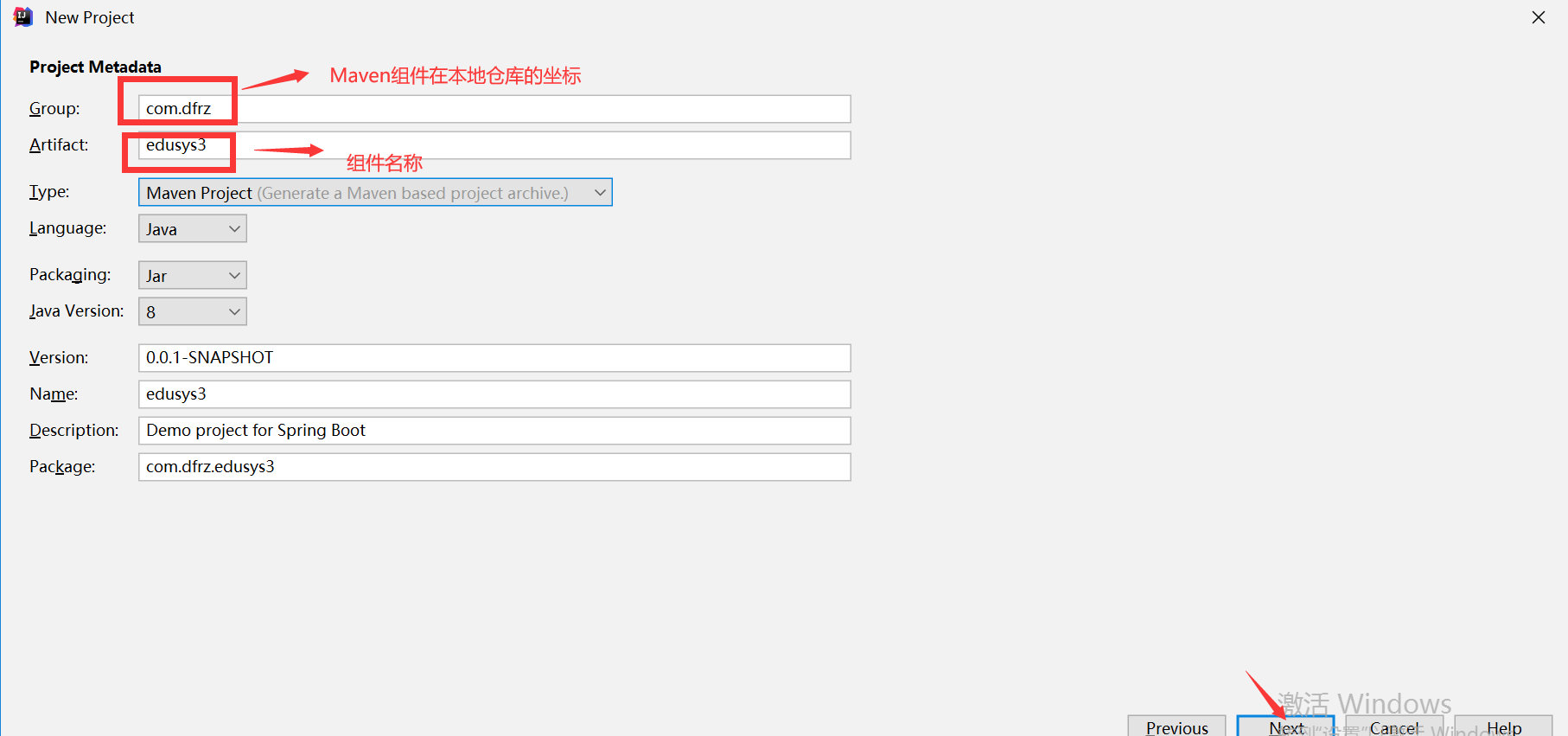


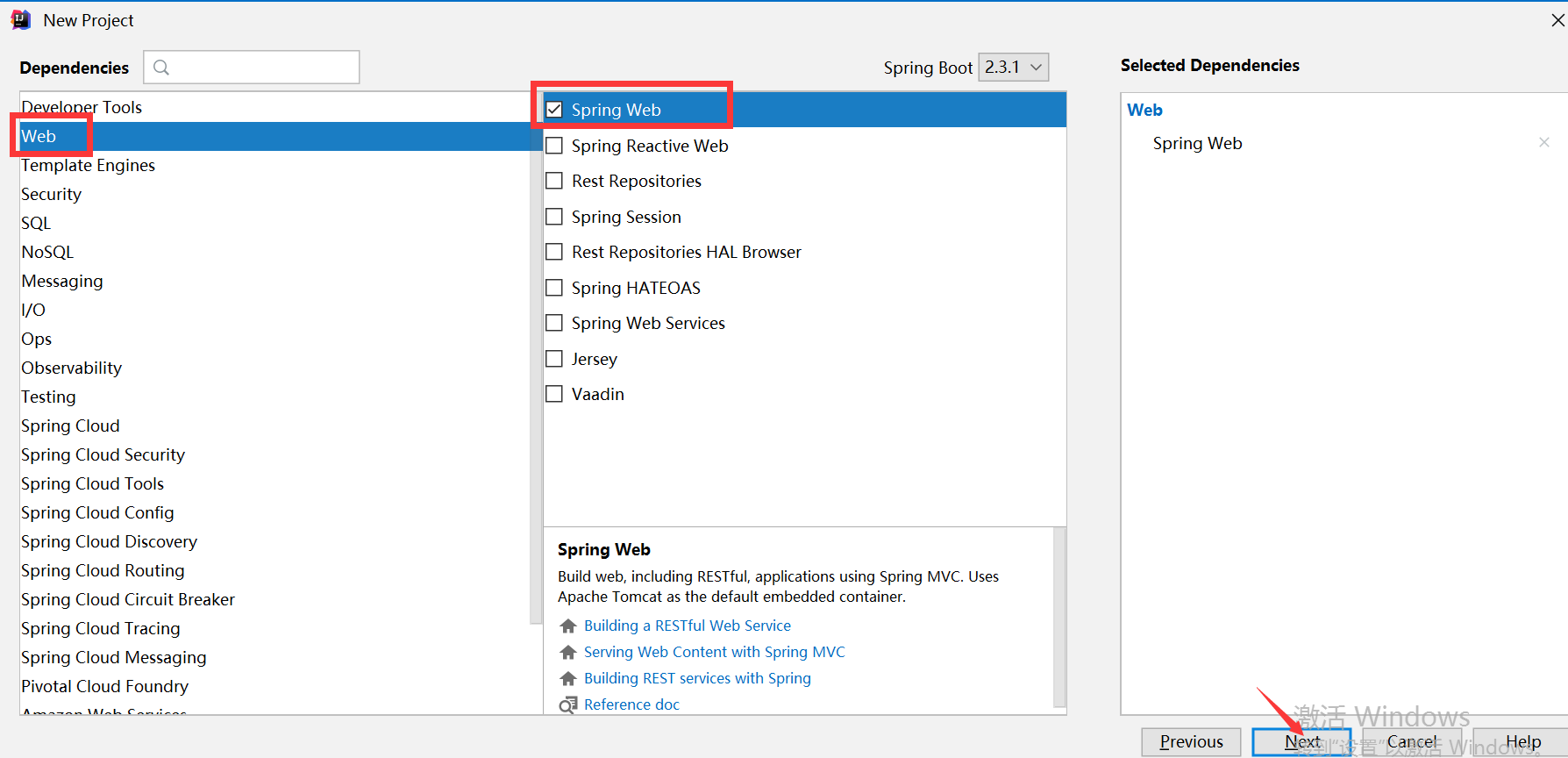


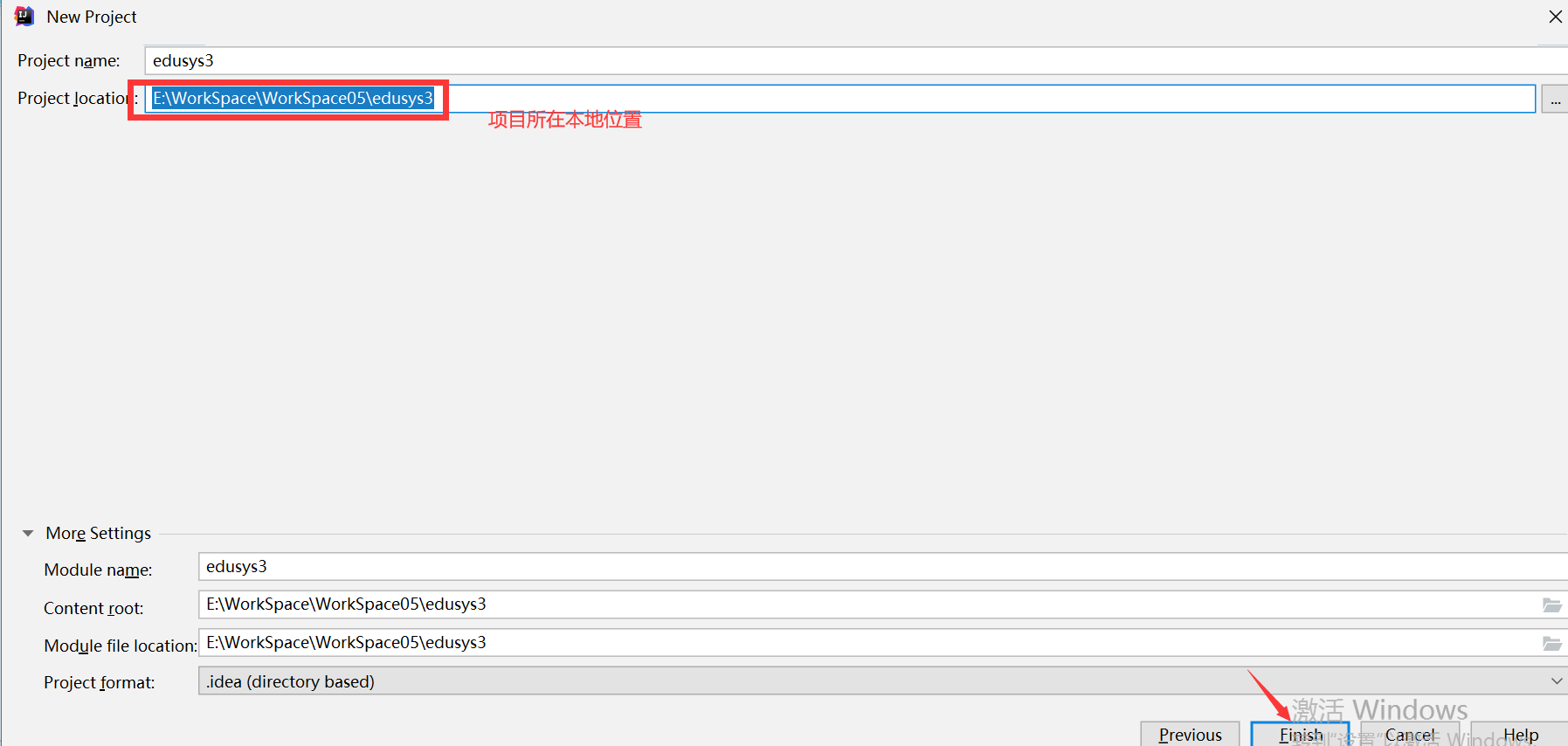
新建SpringBoot项目（基于Maven）

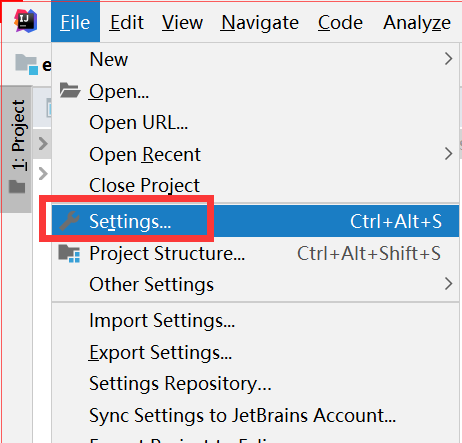


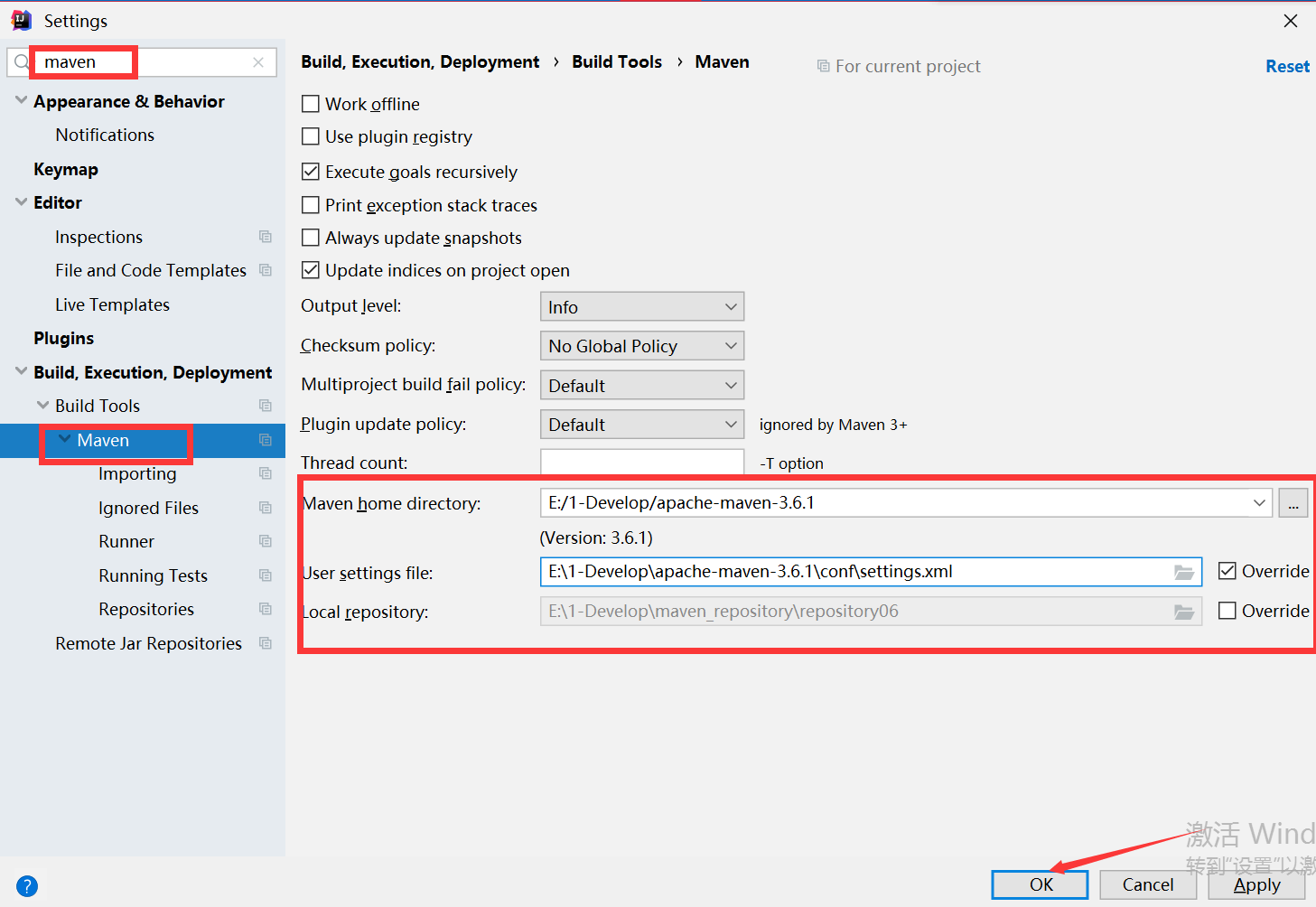


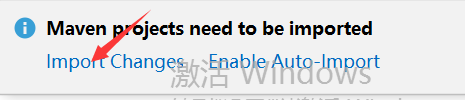












编写一个Spring的依赖注入Demo

步骤1：编写User类

**package** com.dfrz.edusys3.bean;  
  
*/\*用户类\*/***public class** User {  
 *//主键* **private** Integer **id**;  
 *//用户名* **private** String **uname**;  
 *//密码* **private** String **upass**;  
  
 **public** User() {  
 }  
  
 **public** User(Integer id, String uname, String upass) {  
 **this**.**id** = id;  
 **this**.**uname** = uname;  
 **this**.**upass** = upass;  
 }  
  
 setter/getter  
}

**步骤2：编写IUserDao接口以及实现类**

**import** com.dfrz.edusys3.bean.User;  
  
**import** java.util.List;  
  
*/\*\*  
 \* 用户DAO接口  
 \*/***public interface** IUserDao {  
 **public void** save(User user);  
 **public** User getUserById(Integer id);  
 **public** List<User> getUsers();  
}

**package** com.dfrz.edusys3.dao;  
  
**import** com.dfrz.edusys3.bean.User;  
**import** org.springframework.stereotype.Repository;  
  
**import** java.util.List;  
@Repository(**"mybatis"**)  
**public class** UserDaoImpl **implements** IUserDao {  
 @Override  
 **public void** save(User user) {  
 System.***out***.println(**"【添加用户mybatis】"**+user.getUname());  
 }  
  
 @Override  
 **public** User getUserById(Integer id) {  
 System.***out***.println(**"【获取用户mybatis】"**+id);  
 **return null**;  
 }  
  
 @Override  
 **public** List<User> getUsers() {  
 **return null**;  
 }  
}

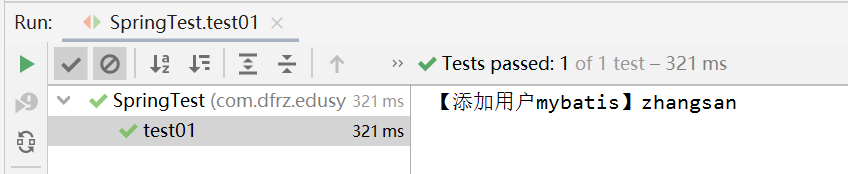
**步骤3：编写IUserService接口以及实现类**

**package** com.dfrz.edusys3.service;  
  
**import** com.dfrz.edusys3.bean.User;  
  
**public interface** IUserService {  
  
 **public void** addUser(User user);  
}

**package** com.dfrz.edusys3.service;  
  
**import** com.dfrz.edusys3.bean.User;  
**import** com.dfrz.edusys3.dao.IUserDao;  
**import** org.springframework.beans.factory.annotation.Autowired;  
**import** org.springframework.beans.factory.annotation.Qualifier;  
**import** org.springframework.stereotype.Service;  
  
*//@Service 注解 ：UserServiceImpl如果打上@Service，  
// 在Spring启动的时候，会被扫描到Spring容器中，组件被Spring容器所管理。*@Service  
**public class** UserServiceImpl **implements** IUserService {  
 *//传统模式 Service 和 DAO 耦合了  
 //Spring的注入方式就是来解决 组件和组件之间的耦合性问题  
 //@Autowired 自动装配的注入方式 ：寻找实现IUserDao接口的实现类* @Autowired  
   
 IUserDao **userDao**;  
 @Override  
 **public void** addUser(User user) {  
 **userDao**.save(user);  
 }  
}

**步骤4：编写单元测试**

**package** com.dfrz.edusys3.test;  
  
**import** com.dfrz.edusys3.bean.User;  
**import** com.dfrz.edusys3.service.IUserService;  
**import** org.junit.Test;  
**import** org.junit.runner.RunWith;  
**import** org.springframework.beans.factory.annotation.Autowired;  
**import** org.springframework.boot.test.context.SpringBootTest;  
**import** org.springframework.test.context.junit4.SpringRunner;  
  
@RunWith(SpringRunner.**class**)  
@SpringBootTest  
**public class** SpringTest {  
 @Autowired  
 IUserService **userService**;  
 @Test  
 **public void** test01(){  
 **userService**.addUser(**new** User(1,**"zhangsan"**,**"333"**));  
 }  
}



工程结构

