# 第六章：Spring核心概念

## Spring概述

### 概念

Spring是开源的轻量级框架，目标是让现有的技术更加易用

Spring框架是一站式框架，Spring在JavaEE三层结构中，每一层都提供不同的解决技术

1. web层：spring mvc
2. service层：控制反转IoC
3. dao层：spring的jdbcTemplate

### Spring两大核心

AOP：面向切面编程

IoC：控制反转，将控制从代码本身转移到web容器

### 优势

* **方便解耦，简化开发**

Spring 就是一个大工厂，可以将所有对象创建和依赖关系维护，交给 Spring 管理

* **AOP 编程的支持**

Spring 提供面向切面编程，可以方便的实现对程序进行权限拦截、运行监控等功能

* **声明式事务的支持**

只需要通过配置就可以完成对事务的管理，而无需手动编程

* **方便程序的测试**

Spring 对 Junit4 支持，可以通过注解方便的测试 Spring 程序

* **方便集成各种优秀框架**

不排斥各种优秀的开源框架，其内部提供了对各种优秀框架（如：Struts、Hibernate、

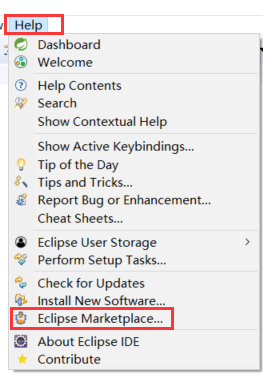
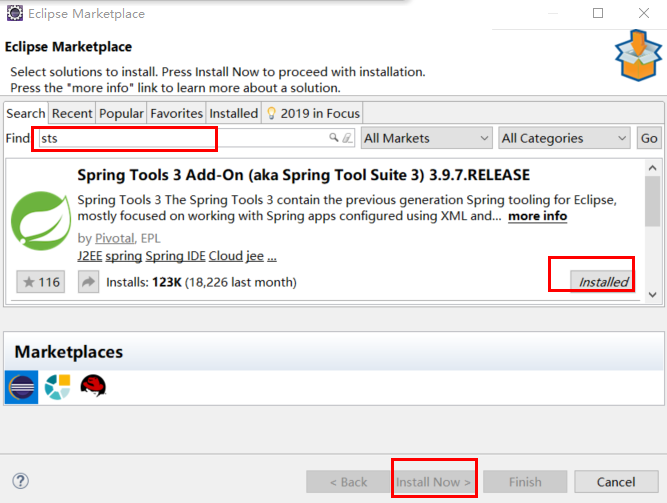
MyBatis、Quartz 等）的直接支持

* **降低 JavaEEAPI 的使用难度**

Spring 对 JavaEE 开发中非常难用的一些 API（JDBC、JavaMail、远程调用等），都提供了封装，

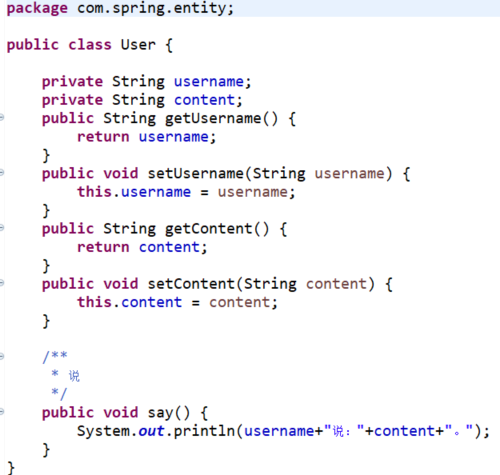
使这些 API 应用难度大大降低

## STS插件

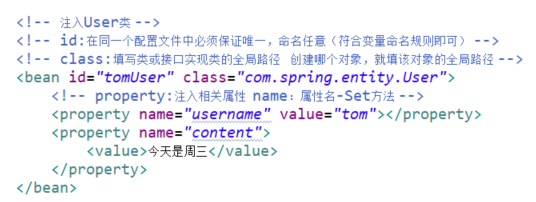
 

## Spring的使用

### 创建User实体类



### 编写applicationContext.xml文件



### 测试类

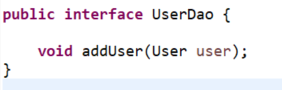


## 模拟添加用户

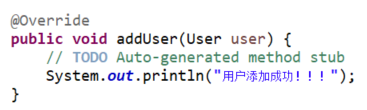
### 创建实体类

同上案例

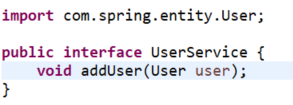
### 创建UserDao接口



### 创建UserDaoImpl实现类



### 创建UserService接口



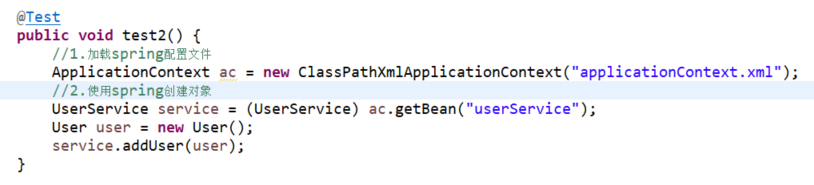
### 创建UserServiceImpl接口



### applicationContext配置文件



### 测试类



## 控制反转IoC

### 概念

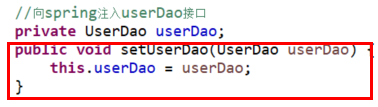
1.控制反转(IoC)也称依赖注入（DI），控制权从代码程序本身转交给spring容器管理

2.组件之间以配置文件的形式组织在一起，而不是以硬编码的方式耦合在一起

### 使用注意事项

控制反转依赖注入的实现方式常用的有set注入、构造注入

Set注入：需要提供set方法，并在spring配置文件中配置（根据set方法注入）





## 面向切面AOP

### 概念

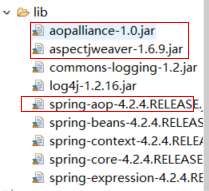
从系统中分离出切面，独立于业务逻辑实现，在程序执行时织入程序中运行（**在什么位置，执行什么功能**）

### 原理

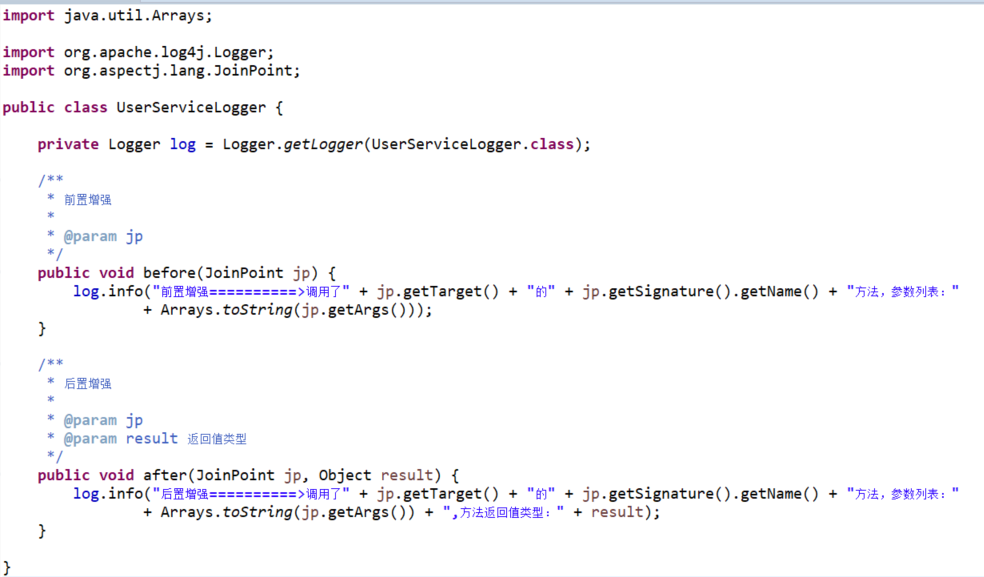
通过动态代理模式在不修改源代码的情况下添加新的功能

### 增强处理

#### 添加aop的jar文件



#### 定义增强类



#### Spring配置文件

