**Spring源码分析**

# 一切从bean开始

## Spring设计目的

Spring是为了简化企业负责应用而生。基本理念是：**简化开发**

**简化开发的策略：**

* 基于POJO的轻量级和最小侵入性编程。
* 通过依赖注入和面向接口编程。
* 基于切面和惯性进行声明式编程。
* 通过切面和模板减少样板式代码。

**总结来说：**

**就是通过面向bean、依赖注入以及面向切面来完成。**

## 面向Bean编程

Spring是面向bean编程的，它的作用相当于Object对于OOP编程的重要性，没有bean就没有Spring。Spring通过IOC容器来管理我们的bean。

## 依赖注入

那么Spring管理我们的bean，自然就为了我们能方便的使用Bean，

**Spring实现的依赖注入：**

1. @Autowrite private InterfaceA a; //可以通过接口找到实现类进行注入。
2. @Resource(“aaa”) A a; //可以通过容器中id自动注入
3. @Autowrite A a; //可以通过类型自动注入

**除了通过注解注入，Spring还提供了其他的注入方式：**

1. setter注入
2. 通过构造方法注入
3. 强制注入

## 面向切面

面向切面编程，即AOP是一种编程思想，它允许程序员对横切关注点或者横切典型的职责分界线的行为(例如日志和事物管理)进行模块化。AOP的核心构造是方面(切面)，它将那些影响多个类的行为封装到可重用的模块

**AOP在Spring中的应用：**

* + - **Authentication 权限认证**
    - **Logging 日志**
    - **Transactions Manager 事物管理**
    - **Lazy Loading 懒加载**
    - **Context Process 上下文处理**
    - **Error Handler 错误跟踪(异常捕获)**
    - **Cache 缓存处理**

# 设计模式

## 代理模式

简单点说就是自己不想做或者没时间做的事情交给别人去做，别人需要获取你自己的信息

**从技术角度来说：**

1. 代理类可以控制目标对象的访问控制。
2. 可以提供更灵活目标对象调用，比如可以在目标对象方法调用之前或者之后做点什么，其实就是方法增强，但是又不会侵入目标对象的代码。
3. 解耦，

**特点：**

1. 有一个代理类和目标类，代理类中有目标类的引用

### 静态代理