# Spring概述

## spring简介

Spring是一个开放源代码的设计层面框架，他解决的是业务逻辑层和其他各层的松耦合问题，因此它将面向接口的编程思想贯穿整个系统应用。Spring是于2003 年兴起的一个轻量级的Java 开发框架，由Rod Johnson创建。简单来说，Spring是一个分层的JavaSE/EE **full-stack(一站式)** [轻量级](https://baike.baidu.com/item/%E8%BD%BB%E9%87%8F%E7%BA%A7/10002835)开源框架。

## Spring与解耦合

Spring根据功能的不同，将代码划分为两类：主业务逻辑与系统级服务（交叉业务逻辑）。主业务逻辑间，及主业务逻辑与系统级服务间的耦合度是较高的。Spring对于前述两种较高的耦合度，采用了两种不同的技术进行解耦。使用IoC解耦主业务逻辑间的耦合度，使用AOP解耦主业务逻辑与系统级服务间的耦合度。

## Spring框架的下载

下载地址：<http://repo.spring.io>

官网： <http://spring.io>

## 第一个Spring程序

# Spring与IoC

IoC，Inversion of Control，控制反转。将传统上由程序代码直接操控的对象创建权交给容器，通过容器来管理对象的生命周期。控制反转是对对象控制权的转移，从代码本身转移到了容器。

IoC是一种思想，是一个概念，其实现方式有很多。当前比较流行的实现方式有两种：依赖注入与依赖查找。

* 依赖查找：DL，Dependency Lookup，代码中需要给出目标对象的存放路径，系统会根据路径到容器中查找相应的对象。典型的应用是JNDI（Java Naming and Directory Interface，Java命名与目录接口，这是一个第三方容器，是一个服务器）。
* 依赖注入：DI，Dependency Injection，代码中无需给出目标对象的定位路径，只需给定对象名称或类型，系统就会直接到容器中进行查找。是目前最优秀的解耦方式。典型应用是Spring。

## 基于XML的DI

### 注入分类

#### 设值注入

#### 构造注入

### 命名空间注入

#### p命名空间设值注入

#### c命名空间构造注入

### 数组与集合属性注入

### 域属性的自动注入

### SPEL注入

### 为应用指定多个Spring配置文件

#### 平等关系的配置文件

#### 包含关系的配置文件

## 基于注解的DI

### 搭建环境

#### 导入aop的Jar包

#### 在xml文件中导入context约束

### 注解

#### @Component

与该注解功能相同，意义不同的注解还有三个：

* @Repository：注解在Dao实现类上
* @Service：注解在Service实现类上
* @Controller：注解在SpringMVC的处理器类

这四个注解都表示，将当前类的对象交给Spring容器来管理。

#### @Value

其在注入值时即没有使用设值注入，也没有使用构造注入，而就是注解注入。即，与set方法、带参构造器没有任何关系。

#### 域属性的自动注入

* @Autowired
* @Qualifier
* @Reource

### CodeConfig容器类

### 使用Spring的JUnit4测试Spring

#### 需要导入spring-test的Jar包

#### 在测试类头上添加两个注解

#### 在测试类中结合@Autowired使用

### 注解与XML共同使用时的优选级

XML的优选级要高于注解的。

# Spring与AOP

## AOP概述

### AOP简介

在软件业，AOP为Aspect Oriented Programming的缩写，意为：[面向切面编程](https://baike.baidu.com/item/%E9%9D%A2%E5%90%91%E5%88%87%E9%9D%A2%E7%BC%96%E7%A8%8B/6016335)，通过[预编译](https://baike.baidu.com/item/%E9%A2%84%E7%BC%96%E8%AF%91/3191547)方式和运行期动态代理实现程序功能的统一维护的一种技术。AOP是[OOP](https://baike.baidu.com/item/OOP)的延续，是软件开发中的一个热点，也是[Spring](https://baike.baidu.com/item/Spring)框架中的一个重要内容，是[函数式编程](https://baike.baidu.com/item/%E5%87%BD%E6%95%B0%E5%BC%8F%E7%BC%96%E7%A8%8B/4035031)的一种衍生范型。利用AOP可以对业务逻辑的各个部分进行隔离，从而使得业务逻辑各部分之间的[耦合度](https://baike.baidu.com/item/%E8%80%A6%E5%90%88%E5%BA%A6/2603938)降低，提高程序的可重用性，同时提高了开发的效率。

AOP底层采用两种动态代理模式实现：JDK的动态代理，与CGLIB动态代理。

### AOP编程术语

#### 切面（Aspect）

交叉业务逻辑，例如事务、安全、日志等，称为切面。

#### 目标对象（Target）

业务类的对象，称为目标对象。

#### 织入（Weaving）

将切面插入到目标对象的目标方法的过程，称为织入。

#### 连接点（JoinPoint）

目标对象中可以被切面织入的方法。

#### 切入点（Pointcut）

目标对象中真正被切面织入的方法。切入点一定是连接点，但连接点不一定是切入点。被标记为final的方法是不能作用连接点与切入点的。

#### 通知（Advice）

通知是切面的一种，可以完成简单的织入功能。通知可以定义切面织入的时间点，切入点定义了切面织入的位置。

#### 顾问（Advisor）

顾问是切面的一种，能够将通知以更为复杂的方式织入到目标对象中，是将通知包装为更复杂的切面的装配器。

## AspectJ对AOP的实现

### AspectJ简介

AspectJ是一个面向切面的框架，它扩展了Java语言。AspectJ定义了AOP语法，所以它有一个专门的[编译器](https://baike.baidu.com/item/%E7%BC%96%E8%AF%91%E5%99%A8)用来生成遵守Java字节编码规范的Class文件。

### AspectJ的通知类型

常用通知有五种：

* 前置通知
* 后置通知
* 环绕通知
* 异常通知
* 最终通知

### AspectJ的切入点表达式

通过切入点表达式可以过滤和选择切入点。

execution(

访问权限类型

返回值类型

全限定性类名

方法名(参数列表)

抛出的异常类型

)

表达式：execution(public \* \*..\*.\*(..))

意义：指定任意公共方法均为切入点。

表达式：execution(\* com.abc.service.\*.\*(..))

意义：指定com.abc.service包中的何意类的任意方法均为切入点。

表达式：execution(\*\*..service.\*.\*(..))

意义：指定service包（该包前可以有多级父包）中的何意类的任意方法均为切入点。

* com.abc.service 是匹配的
* com.service 是匹配的
* service 是匹配的

表达式：execution(\*\*.service.\*.\*(..))

意义：指定service包（该包前只能有一级父包）中的何意类的任意方法均为切入点。

* com.abc.service 是不匹配的
* com.service 是匹配的
* service 是不匹配的

表达式：execution(\*\*..\*.doSome(..))

意义：指定何意类的doSome()方法均为切入点。

## AspectJ的开发环境搭建

### 导入依赖

#### AspectJ核心Jar包

#### Spring整合AspectJ的Jar包

#### Spring对AOP的实现Jar包

#### AOP联盟的Jar包

### 修改Spring配置文件

#### Spring配置文件添加AOP约束

#### 注册AOP的AspectJ自动代理

## AspectJ基于注解的AOP实现

* 前置通知
* 后置通知
* 环绕通知
* 异常通知
* 最终通知

## AspectJ基于XML的AOP实现

* 前置通知
* 后置通知
* 环绕通知
* 异常通知
* 最终通知

# Spring与DAO

## Spring与MyBatis整合

这是典型的IoC的应用。

### 导入依赖

#### Spring的5个基本Jar包

#### Spring的JDBC的Jar包

#### Spring事务Jar包

#### spring的AOP的Jar包

#### mybatis的核心Jar包

#### mybatis与Spring整合的Jar包

#### mysql驱动Jar包

#### Druid数据源Jar包

#### log4j的Jar包

#### slf4j的Jar包

#### slf4j与log4j整合Jar包

### 编写代码

#### 定义实体类与DB表

#### 定义Service接口及实现类

#### 定义Dao接口

#### 定义测试类

### 编写配置文件

#### 定义MyBatis的两个配置文件

#### 定义Spring相关配置文件

## Spring与事务

这是典型的AOP的应用。

### Spring事务管理的API

#### 平台事务管理器接口

##### 常用的实现类

* DataSourceTransactionManager：使用JDBC、MyBatis、iBatis等进行持久化时使用。
* HibernateTransactionManager：使用Hibernate进行持久化时使用。

##### Spring回滚方式

Spring事务默认的回滚方式是：发生运行时异常回滚，发生受查异常提交。

#### 事务定义接口

MySQL默认的事务隔离级别是可重复读；Oracle默认的事务隔离级别是读已提交。

事务传播行为是指，处于不同事务中的方法在相互调用时，执行期间事务的维护情况。

### 使用Spring的事务注解管理事务

### 使用AspectJ管理事务

# Spring与Web

## 不使用Spring的web依赖

## 使用Spring的web依赖

首先需要导入spring-web依赖。