目录

[Day24 2017-10-27 2](#_Toc496868394)

[Spring引入properties文件 2](#_Toc496868395)

[Spring IOC和AOP 2](#_Toc496868396)

[IOC 控制反转（Inversion of Controller） 2](#_Toc496868397)

[AOP面向切面编程（Aspet Oriented Programming） 4](#_Toc496868398)

# Day24 2017-10-27

## Spring引入properties文件

<http://blog.csdn.net/eson_15/article/details/51365707>

1. Xml文件中引用

<context:property-placeholder location=””/>

如：

<**context:property-placeholder** location="db-config.properties"/>

或者<bean/>标签

如：

<!-- 用于读取properties文件，此处将数据库连接的用户名密码等存在db-config.properties文件中 -->  
<**bean** id="propertyConfigurer"  
 class="org.springframework.beans.factory.config.PropertyPlaceholderConfigurer">  
 <**property** name="locations">  
 <**list**>  
 <**value**>db-config.properties</**value**>  
 </**list**>  
 </**property**>  
</**bean**>

1. 注解方式加载

## Spring IOC和AOP

<http://blog.csdn.net/luoshenfu001/article/details/5816408/>

### IOC 控制反转（Inversion of Controller）

1. 深入理解IOC和DI（依赖注入）

<http://www.importnew.com/13619.html>

IOC和DI中包含的面向对象思想：开闭原则（O），里氏替换原则（L），依赖倒置原则（D），迪米特原则。

普通的编程模式：

public class A {

private InterfaceB cb;

public A(){

cb = new ClassBImplementsInterfaceB();

}

public void method(){

cd.doThings();

}

}

其中InterfaceB接口定义了doThings方法。此时类A的依赖情况为：

A即依赖于接口（InterfaceB），又同时依赖实现（ClassBImplementsInterfaceB），**因为它需要在编译阶段就确定使用哪种实现。**

**使类A对于选择哪个具体InterfaceB实现类的权利从其中移除，转交给外部决定（Spring）（控制反转）。**即它所依赖的对象由外部注入（决定）**（依赖注入）**

1. 传统编程和IoC的对比

传统编程：决定使用哪个具体的实现类的控制权在类本身，在编译阶段就确定了。

IoC模式：调用类只依赖接口，而不依赖具体的实现类，减少了耦合。控制权交给了容器，在运行的时候才由容器决定将具体的实现动态的“注入”到调用类的对象中。

1. IOC目的

降低类间耦合，增强代码复用性；

面向接口编程，实施依赖倒置原则；

提高系统可插入、可测试和可修改的特性。

1. Spring中的IOC

IOC容器负责容纳bean，并对bean进行管理，在Spring中，BeanFactory是IOC的核心接口，其职责包括：实例化、定位、配置引用程序中的对象，并建立这些对象间的依赖。常见的BeanFactory有XMLBeanFactory。

### AOP面向切面编程（Aspet Oriented Programming）