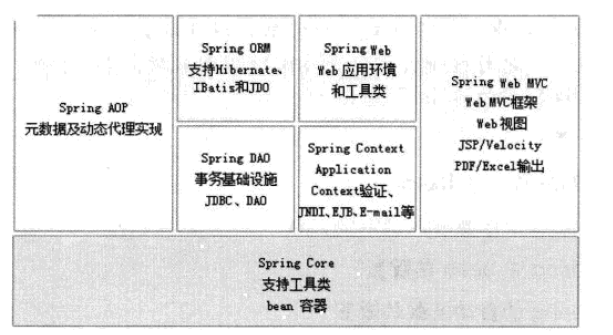
主要工程php开发，曾经看过一些，但因为很少上手去操作，基本也都忘了。

对spring 7大模块有概念性的理解，像spring mvc,dao,orm这些都是框架通行的逻辑。对于特色Ioc,AOP的原理有些理解就是了。



**核心容器(Spring core)**

核心容器提供spring框架的基本功能。Spring以bean的方式组织和管理Java应用中的各个组件及其关系。Spring使用BeanFactory来产生和管理Bean，它是工厂模式的实现。

Spring上下文(Spring context)

构建于Core封装包基础上的 Context封装包，提供了一种框架式的对象访问方法，有些象JNDI注册器。Context封装包的特性得自于Beans封装包，并添加了对国际化（I18N）的支持（例如资源绑定），事件传播，资源装载的方式和Context的透明创建，比如说通过Servlet容器。

**Spring面向切面编程(Spring AOP)**

通过配置管理特性，Spring AOP 模块直接将面向方面的编程功能集成到了 Spring框架中。所以，可以很容易地使 Spring框架管理的任何对象支持 AOP。Spring AOP 模块为基于 Spring 的应用程序中的对象提供了事务管理服务。通过使用 Spring AOP，不用依赖 EJB 组件，就可以将声明性事务管理集成到应用程序中。

**Spring DAO模块**

DAO模式主要目的是将持久层相关问题与一般的的业务规则和工作流隔离开来。Spring 中的DAO提供一致的方式访问数据库，不管采用何种持久化技术，Spring都提供一直的编程模型。Spring还对不同的持久层技术提供一致的DAO方式的异常层次结构。

**Spring ORM模块**

Spring 与所有的主要的ORM映射框架都集成的很好，包括hibernate、JDO实现、TopLink和IBatis SQL Map等。Spring为所有的这些框架提供了模板之类的辅助类，达成了一致的编程风格。

**Spring Web模块**

Web上下文模块建立在应用程序上下文模块之上，为基于Web的应用程序提供了上下文。Web层使用Web层框架，可选的，可以是Spring自己的MVC框架，或者提供的Web框架，如Struts、Webwork、tapestry和jsf。

**Spring MVC框架(Spring WebMVC)**

MVC框架是一个全功能的构建Web应用程序的MVC实现。通过策略接口，MVC框架变成为高度可配置的。Spring的MVC框架提供清晰的角色划分：控制器、验证器、命令对象、表单对象和模型对象、分发器、处理器映射和视图解析器。Spring支持多种视图技术

详解：

Ioc:

理解和介绍：

Inversion of control,控制反转是关于一个对象如何获取他所依赖的对象的引用，这个责任的反转

控制反转一般分为两种类型，依赖注入（Dependency Injection，简称DI）和依赖查找（Dependency Lookup）。依赖注入应用比较广泛。

ioc是目的，di是手段。ioc是指让生成类的方式由传统方式（new）反过来，既程序员不调用new,需要类的时候由框架注入（di）。

更多：see-17.2.2

AOP:面向切面编程

作用：**将业务逻辑和通用逻辑（如log,权限验证等）进行分离，将通用逻辑视为一个切面织衣入到业务逻辑的某些**。

AOP把软件系统分为两个部分：核心关注点和横切关注点。

业务处理的主要流程是核心关注点，与之关系不大的部分是横切关注点。

**横切关注点的一个特点是，他们经常发生在核心关注点的多处，而各处都基本相似。比如权限认证、日志、事务处理**。

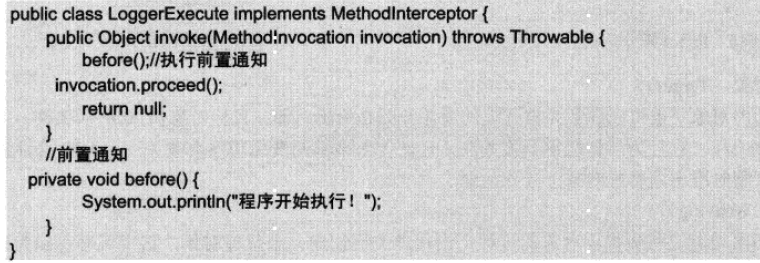
角色:切入点、通知（拦截对象目标的方法，执行切面）

原理：动态代理：jdk动态代理或cglib

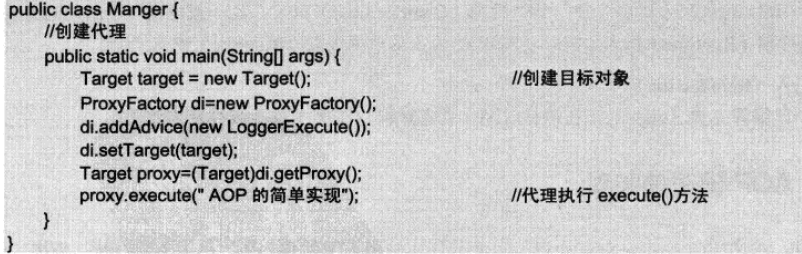
<http://blog.csdn.net/moreevan/article/details/11977115/>

简单实现：

通知：



调用



原理：**将切面使用动态代理的方式动态织入到目标对象，形成一个代理对象**。**ProxyFactory本质是生成一个代理对象**，代理对象为Target,在invoke中执行method.invoke()时会相应执行loggerExecute的before等作为通知链，由此实现切入。

**InvocationHandler**的invoke主流程可以简述为：获取可以应用到此方法上的通知链（Interceptor Chain）,如果有,则应用通知,并执行joinpoint; 如果没有,则直接反射执行joinpoint。然后执行method.invoke(target, args)

生成代理方法的流程：

**0）验证对象相应方法是否存在**

**1）获取拦截器**

**2）判断拦截器链是否为空（会使用缓存），如果是空的话直接调用切点方法**

**3）如果拦截器不为空的话那么便创建ReflectiveMethodInvocation类，把拦截器方法（还有原方法）都封装在里面，然后执行**

作业帮aop，

Class xx extends advisor{

invoke(){

//实际逻辑

}

}

Advisor {

Execute(){ //通知，通过通知执行切面

//前置逻辑——如打点，用户信息，请求过滤

Invoke();

//后置逻辑

}

}

另外每个action层都要有自己的conf文件，不是xml，用于做参数验证，登陆验证等操作。

Spring提供其中三种切入点，静态、动态和其他切入点。

**静态切入点**

只限于给定的方法和目标类，而不考虑方法的参数。Spring在调用切入点时只在第一次的时候计算静态切入点的位置，然后把它**缓存**起来，以后就不需要在进行计算。

**动态切入点**

动态切入点与静态切入点的区别是，它不仅限定于给点的方法和类，动态切入点还可以以指定方法的参数。因为参数的变化性，**所有动态切入点不能缓存**，需要么次调用的时候都进行计算，因此使用动态切入点有很大性能损耗。

例子：

<beanid="settersAndAbsquatulatePointcut" class="org.springframework.aop.support.RegexpMethodPointcut">

<property name="patterns">

<!--设定切入点>

<list>

<value>.\*save.\*</value>

</list>

</porperty>

</bean>

以正则形式匹配：

.\*save.\*：表示所有以save开头的方法都是切入点

DAO/ORM

对象关系映射（Object Relational Mapping）

关系型数据库和业务实体对象之间作一个映射，这样，我们在具体的操作业务对象的时候，就不需要再去和复杂的SQL语句打交道，只需简单的操作对象的属性和方法。

用object来map我们的db

Spring mvc

MVC

模型(model)－视图(view)－控制器(controller)的缩写

Model（模型）是应用程序中用于处理应用程序数据逻辑的部分。 通常模型对象负责在**数据库中读取数据**。

View（视图）是应用程序中处理**数据显示**的部分。

Controller（控制器）是应用程序中处理用户交互的部分。 通常控制器负责从视图读取数据，控制用户输入，并向模型发送数据。