IOC

1. Spring注入方式 : setter注入(最常用), 构造方法注入, 接口注入
   注入形式 : XML注入, 注解注入
2. bean命名方式 : id和name(id不可重复, name可以包含特殊字符)
3. 简单属性注入 : <property name="xxx" value="ooo"/>
4. bean的作用域scope : singleton, prototype
5. <list><value>ListA</value><value>ListB</value></list>
   <map><entry key="key1" value="value1" /></map>
   <set><value>SetA</value></set>
   <props><prop key="prop1">value1</prop></props>
6. 注解: @Scope指定作用域

   <context:component-scan/>包含<context:annotation-config/>的功能
   并扫描类的@Component @Controller @Service @Repository注解

   <context:annotation-config/>某bean配置在applicationcontext.xml里, 需要
   @AutoWired(默认byType,不需要写setter方法)
   @Resource(默认byName,不需要写setter方法)注入的

   @PostConstruct = init-method
   @PreDestroy = destroy-method
7. bean的生命周期.....

AOP

servlet的filter和Spring/Struts2的Intercepter都是AOP的实现, AOP的原理是动态代理

用途 : 访问控制、事务管理、日志记录、缓存、对象池管理
<aop:aspcetj-autoproxy /> spring容器启动过程中,
只要发现有需要产生代理的就自动产生代理

aspcetj是专门面向切面编程的框架, spring使用了aspectj

1. Annotation方式:
<aop:aspcetj-autoproxy />
@Aspect
JoinPoint

PointCut("excution (public * com.hexun..*.*(..))") 路径的集合, 可以代替@Before的路径

@Before advice[@Before("excution (public * com.hexun..*.*(..))")]

@After advice(相当于try/catch/finally的finally)

@AfterReturning advice(有返回值的时候, 先After后AfterReturning)

@AfterThrowing advice(相当于try/catch/finally的catch)

@Around advice (责任链设计模式)

target

weave织入

2.XML方式:

<aop:confing>

<aop:pointcut>

<aop:aspect>

<aop:advisor>

<aop:before><aop:after>.....

```
<aop:config>
    <aop:pointcut expression="execution(public *
com.bjsxt.service..*.add(..))"
                  id="servicePointcut"/>
    <!-- 将容器中的logInterceptor普通Bean转换成切面Bean -->
    <aop:aspect id="logAspect" ref="logInterceptor">
        <aop:before method="before"  pointcut-ref="servicePointcut"
/>
    </aop:aspect>
</aop:config>
```