**Spring**

**Spring容器bean的对象生命周期：**

1、生命周期的开始：单例，由加载完spring容器开始；多例，由从容器获得bean对象开始。2、实例化3、初始化。4、服务。5、销毁(单例：关闭容器的时候，多例由jvm自动回收)

**IoC:**

对象的创建、赋值的代码，反转给IoC容器实现和管理，用来减低代码之间的耦合度（ioc也可以称di）

**AOP：**

在系统开发中，将系统的共性的公共的功能独立实现（封转在一个类中），在程序运行的过程中，将共性功能和核心的业务功能进行整合，用来解除共性功能与核心业务功能间的耦合、提高代码的复用性。

名词解释：

1、关注点：重复代码（安全、事务、日志）

2、切面：关注点封装成的类（通知和切点的集合）

3、连接点：能够插入一个切面的点（如类开始初始化前、类初始化后、类某个方法调用前、调用后、方法抛出异常后）

4、通知：描述切面要完成的工作，何时执行这个工作（是织入到目标类连接点上的一段程序代码）

5、切点：定义通知被应用的位置（一个切点可以匹配多个连接点）

6、引入：不修改代码的前提下，运行期为现有的类动态添加新方法或属性（引入是一种特殊的通知，把切面用到目标类中）

7、织入：将通知添加到目标类具体连接点上的过程（AOP像一台织布机，将目标类、通知或引入通过AOP这台织布机天衣无缝地编织到一起。AOP有三种织入的方式实现技术（Spring采用动态代理织入））：

a、编译期织入，这要求使用特殊的Java编译器。

b、类装载期织入，这要求使用特殊的类装载器。

c、动态代理织入，在运行期为目标类添加增强生成子类的方式。

**静态代理：**

作用：是业务代码与日志代码完全分离，实现松散耦合。

缺点：一个代理接口只能服务于一种类型的对象，对于稍大点的项目根本无法胜任。

代理对象与被代理对象必须实现同一接口，在代理对象中实现与日志记录的相关服务，被代理对象只保留业务代码，并在需要的时候呼叫被代理对象。

**动态代理：**

使用动态代理，可以使得一个处理者(Handler)服务于各个对象，处理者的类设计必须实现java.lang.reflect.InvocationHandler接口（通过InvocationHandler接口实现的动态代理只能代理接口的实现类）

**spring两种代理方式：**

1、若目标对象实现了若干接口，spring使用JDK的java.lang.reflect.Proxy类代理。

优点：因为有接口，所以使系统更加松耦合

缺点：为每一个目标类创建接口

2、若目标对象没有实现任何接口，spring使用CGLIB库生成目标对象的子类。

优点：因为代理类与目标类是继承关系，所以不需要有接口的存在。

缺点：因为没有使用接口，所以系统的耦合性没有使用JDK的动态代理好。

**bean的三种创建方式：**

1. 调用无参数的构造方法：<bean id="user1" class="com.itcast.pojo.User"></bean>
2. 带参数构造器创建对象：

<bean id="user2" class="com.itcast.pojo.User">

<constructor-arg index="0" type="int" value="100"></constructor-arg>

<constructor-arg index="1" type="java.lang.String" value="Jack"></constructor-arg>

</bean>

1. 通过工厂类创建对象：

非静态方法创建对象

<bean id="factory" class="com.itcast.pojo.User"></bean>

<bean id="user3" factory-bean="factory" factory-method="getInstance"></bean>

静态工厂创建对象（获得对象的方法是静态的）

<bean id="user4" class="com.itcast.geeker.ObjectFactory" factory-method="getStaticInstance"></bean>

**spring的自动注入：**

byName：根据属性和组件的名称匹配关系来实现bean的自动装配

byType：根据属性和组件的类型匹配关系来实现bean的自动装配，有多个适合类型的对象时装配失败

constructor：与byType类似是根据类型进行自动装配，但是要求待装配的bean有相应的构造函数

autodetect：利用Spring的自省机制判断使用byType或是constructor装配

<bean id="student" class="com.test.Student" autowire="byName">（注入teacher属性）

<property name="name" value="张三"/>

</bean>

**bean创建：单例/多例 ：**

在bean.xml配置文件中，配置scope=”singleton”，为单例创建对象，如果不配置，默认也是单例方式。主要应用于service/dao层还有一些utils工具类等只需要在服务器启动时初始化一次即可。

scope=”prototype”，表示多例方式创建对象，也就是当用户需要使用这个对象时即创建对象，在ssh项目整合中主要应用于Action对象。这种方式一般在服务器启动时不会创建对象，在每次使用才创建，且整个应用只有一个。

**bean作用范围：**

singleton：单例模式Spring ioc容器中仅有一个Bean实例，Bean以单例的方式存在；

prototype：原型模式（多例模式）每次从容器中调用 Bean 时，都返回一个新的实例；

request：每次HTTP请求都会创建一个新的Bean；

session：同一个HTTP Session 共享一个Bean；

global session：同一个全局Session共享一个Bean，一般用于Portlet应用环境；

application：同一个Application共享一个Bean；

**基于注解的bean定义（不需要提供setter方法）：**

这些注解的作用与XML中定义一样，将组件交给Spring容器管理。组件的名称默认是类名（首字母变小写）

@Component：当对组件的层次难以定位的时候使用这个注解（@Component("teacher")、@Value("李四")、@Resource Teacher teacher）xml中配置<context:component-scan base-package="test，dev，online"/>，@Resource为j2e注解

@Controller：表示控制层的组件

@Service：表示业务逻辑层的组件

@Repository：表示数据访问层的组件

**Spring常用注解：**

@ResponseBody表示该方法的返回结果直接写入HTTP responsebody中，返回结果不会解析为路径。

@Pathvariabl(value=”id”)e绑定@RequestMapping(value=”get/{id}”)映射路径上的参数。

@RequestParam(value=”id”) 该注解可以代替request.getParameter("name")。

@Transactional(propagation = Propagation.REQUIRED, readOnly = false, rollbackFor = Exception.class) 开启声明式事物管理，需要在配置文件中配置<tx:annotation-driven transaction-manager="transactionManager" />。

@Scope(value=“prototype”) 作用域用在类上，每次请求时都会创建一个新的实例。

@Configuration 表示类可以被Spring的IoC容器所使用，作为bean定义的资源。

**SpringMVC**

**SpringMVC优点：**

1. 以DispatchServlet为核心控制器，该控制器负责拦截用户的所有请求，无需自己编写
2. 是Spring子模块，与Spring无缝整合。
3. 使采用MVC设计模式使得各层分工明确，降低耦合。

**SpringMVC配置文件：**

1、<context:annotation-config />注册使用注解前必须使用的bean

2、<context:component-scan />不仅具有以上标签的功能，还可以扫描包。子标签<context:include-filter />用来指定扫描的注解类型。

3、<mvc:annotation-driven />注册默认处理请求的处理器适配器、处理器映射器等。启用注解驱动。

4、<mvc:default-servlet-handler />处理静态资源请求。