1、什么是Spring框架？Spring框架有哪些主要模块？

Spring框架至今已集成了20多个模块。这些模块主要被分如下图所示的核心容器、数据访问/集成,、Web、AOP（面向切面编程）、工具、消息和测试模块

Spring是一个轻量级的IOC和AOP容器框架

a,轻量级：占用资源不是很多，没有侵入性；

b，IOC（Inversion of Control 控制反转）：对象创建责任的反转（重点，核心）；

c, Aop(Aspect Oriented Programming):一种面向横切面编程的思想方式，可以进行功能性扩展d，容器：可以容纳对象，并且可以控制对象的生命周期；

[](http://www.importnew.com/?attachment_id=15854)

2、使用Spring框架能带来哪些好处？

* [**Dependency Injection(DI)**](http://howtodoinjava.com/2013/03/19/inversion-of-control-ioc-and-dependency-injection-di-patterns-in-spring-framework-and-related-interview-questions/) 方法使得构造器和JavaBean properties文件中的依赖关系一目了然。
* Spring可以集成其他框架，如ORM框架等
* 利用JavaBean形式的POJO类，可以很方便的利用依赖注入来写入测试数据。
* Spring MVC提供了对Web的支持

3、什么是控制反转(IOC)？什么是依赖注入？

控制反转：Spring容器来实现相互依赖对象的创建、协调工作。对象只需要关系业务逻辑本身就可以了。从这方面来说，对象如何得到他的协作对象的责任被反转了

IoC的一个重点是在系统运行中，动态的向某个对象提供它所需要的其他对象。这一点是通过DI（Dependency Injection，依赖注入）来实现的。比如对象A需要操作数据库，以前我们总是要在A中自己编写代码来获得一个Connection对象，有了 spring我们就只需要告诉spring，A中需要一个Connection，至于这个Connection怎么构造，何时构造，A不需要知道。在系统运行时，spring会在适当的时候制造一个Connection，然后像打针一样，注射到A当中，这样就完成了对各个对象之间关系的控制。A需要依赖 Connection才能正常运行，而这个Connection是由spring注入到A中的，依赖注入的名字就这么来的。那么DI是如何实现的呢？ Java 1.3之后一个重要特征是反射（reflection），它允许程序在运行的时候动态的生成对象、执行对象的方法、改变对象的属性，spring就是通过反射来实现注入的

在Java中依然注入有以下三种实现方式：构造器注入；Setter方法注入；接口注入

5、BeanFactory和ApplicationContext有什么区别？

BeanFactory 可以理解为含有bean集合的工厂类。产生bean对象

application context如同bean factory一样具有bean定义、bean关联关系的设置，根据请求分发bean的功能。但application context在此基础上还提供了其他的功能，如统一的资源文件读取方式

等

25、Spring 框架中都用到了哪些设计模式？

Spring框架中使用到了大量的设计模式，下面列举了比较有代表性的：

* 代理模式—在AOP和remoting中被用的比较多。
* 单例模式—在spring配置文件中定义的bean默认为单例模式。
* 模板方法—用来解决代码重复的问题。比如. **[RestTemplate](http://howtodoinjava.com/2015/02/20/spring-restful-client-resttemplate-example/" \t "_blank" \o "Spring RESTFul Client – RestTemplate Example)**, JmsTemplate, JpaTemplate。
* 前端控制器—Spring提供了DispatcherServlet来对请求进行分发。
* 视图帮助(**View Helper** )—Spring提供了一系列的JSP标签，高效宏来辅助将分散的代码整合在视图里。
* 依赖注入—贯穿于BeanFactory / ApplicationContext接口的核心理念。
* 工厂模式—BeanFactory用来创建对象的实例。

AOP模块

SpringAOP的实现机制：

Spring AOP采用动态代理机制和字节码生成技术实现

默认情况下，如果Spring AOP发现目标对象实现了相应Interface，则采用动态代理机制为其生成代理对象实例。而如果目标对象没有实现任何Interface，Spring AOP会尝试使用CGLIB(Code Generation Library)这种动态字节码生成类库，为目标对象生成动态的代理对象实例。

DI

public class SimpleIoc {

private static final Map<Class<?>, Object> BEAN\_MAP = new HashMap<Class<?>, Object>();

private static Set<Class<?>> beanClassSet = new HashSet<Class<?>>();

//根据指定包扫描有注解@Bean获取Class

private static void initClassSet() {

for(bean:filesInPackage) {

beanClassSet.add(Class.forName(bean.getClassName()));

}

}

private static void initBeanMap() {

for (Class<?> beanClass : beanClassSet) {

Object obj = Reflecttion.newInstance(beanClass);

BEAN\_MAP.put(beanClass, obj);

}

}

public static void IocInit() {

for(Map.Entry<Class<?>,Object>beanEntry : BEAN\_MAP.entrySet()) {

Class<?> beanClass = beanEntry.getKey();

Object beanInstance = beanEntry.getValue();

//获取Bean定义的所有成员变量

Field[] beanFileds = beanClass.getDeclaredFileds();

for(Field beanField : beanFileds) {

if(beanFiled.isAnnotation()) {//判断是否带有注解，注入其实例对象

Class<?> beanFiledClass = beanFiled.getType();

Object beanFiledInstance = BEAN\_MAP.get(beanFiledClass);

ReflectionUtil.setFiled(beanInstance, beanFiled, beanFiledInstance);

}

}

}

}

}

基本步骤：先遍历包内有Bean注解的类，存到Set里，然后逐个取其Class存到另一个Set里，再逐个取出得到Instance，放到Map里，最后遍历Map，得到Field(通过getDeclaredFileds())，根据Filed从map里找到其实例对象，再把Instanceset set进去，这就是为什么需要有set方法