1. Spring框架
2. 请解释SpringBean的生命周期

Servlet生命周期：实例化，初始化init，接受请求service，销毁destroy

Spring上下文中的Bean生命周期也一样

1、实例化Bean 2、设置对象属性依赖注入 3、处理Aware接口

4、BeanPostProcessor 5、InitializingBean，init-method

6、如果这个Bean 实现了BeanPostProcessor 接口，将会调用

postProcessAfterInitialization(Object obj, String s)方法

7、DisposableBean 8、destroy-method

1. 解释Spring支持的几种bean的作用域
   1. Singleton：默认，每个容器中只有一个bean实例，单例模式由BeanFactory自身维护
   2. Prototype：为每一个bean请求提供一个实例
   3. Request：为每一个网络请求创建一个实例
   4. Session：与request范围类似，确保每个session中有一个bean的实例，在session过期后，bean会随之失效
   5. Global-session：全局作用域，global-session和Protlet应用相关
2. 使用注解之前要开启自动扫描功能
   1. @Scope 注解 作用域
   2. @Lazy(true) 表示延迟初始化
   3. @Service 用于标注业务层组件、
   4. @Controller 用于标注控制层组件（如 struts 中的 action）
   5. @Repository 用于标注数据访问组件，即 DAO 组件。
   6. @Component 泛指组件，当组件不好归类的时候
   7. @Scope 用于指定 scope 作用域的（用在类上）
   8. @Resource 默认按名称装配
   9. @DependsOn：定义 Bean 初始化及销毁时的顺序
   10. @Primary：自动装配时当出现多个 Bean 候选者时
   11. @Autowired 默认按类型装配
   12. @Autowired @Qualifier("personDaoBean") 存在多个实例配合使用
3. Spring 框架中的单例 Beans 是线程安全的么？

Spring框架并没有对单例 bean 进行任何多线程的封装处理，bean的线程安全和并发问题需要开发者自行搞定，大部分springbean没有可变的状态，所以在某种状态上说spring的单例bean是线程安全的。

5、Spring 如何处理线程并发问题？

无状态的 Bean 才可以在多线程环境下共享，在Spring 中，Bean 都可以声明为 singleton作用域，Spring对一些 Bean中非线程安全状态采用 ThreadLocal进行处理，解决线程安全问题。

ThreadLocal和线程同步机制都是为了解决多线程中相同变量的访问冲突问题

ThreadLocal会为每一个线程提供一个独立的变量副本，从而隔离了多个线程对数据的访问冲突

* 1. Spring 基于 xml 注入 bean 的几种方式：

Set方法注入、构造器注入、静态工厂注入、实例工厂

* 1. Spring 的自动装配：

在spring中，对象无法自己查找创建与其关联的其他对象，由容器负责把需要相互协作的对象引用赋予各个对象，使用autowire来配置自动装载

五种自动装配

1. no默认不进行自动装配
2. byname：通过bean的名称进行自动装配
3. byType：通过参数的数据类型进行自动装配
4. constructor：利用构造函数进行装配
5. autodetect：自动探测，如果有构造方法，通过construct自动装配

8、@Autowired 和@Resource 之间的区别

(1) @Autowired默认是按照类型装配注入的，默认情况下它要求依赖对象必须存

在（可以设置它required属性为 false）。

(2) @Resource默认是按照名称来装配注入的，只有当找不到与名称匹配的 bean

才会按照类型来装配注入。

9、Spring 框架中都用到了哪些设计模式？

1、工厂模式：BeanFactory就是简单工厂模式的体现，用来创建对象的实例；

2、单例模式：Bean 默认为单例模式。

3、代理模式：Spring 的AOP 功能用到了 JDK 的动态代理和 CGLIB字节码生成技术；

4、模板方法：用来解决代码重复的问题。比如. RestTemplate, JmsTemplate, JpaTemplate。

5、观察者模式：定义对象键一种一对多的依赖关系，当一个对象的状态发生改变时，所有依赖于它的对象都会得到通知被制动更新，如Spring中listener 的实现--ApplicationListener。

10、Spring 事务的实现方式和实现原理：

（1）Spring事务的种类：

①编程式事务管理使用TransactionTemplate。

②声明式事务管理建立在AOP 之上的。

其本质是通过 AOP 功能，对方法前后进行拦截，将事务处理的功能编织到拦截的方中，也就是在目标方法开始之前加入一个事务，在执行完目标方法之后根据执行情况提交或者回滚事务

（2）spring的事务传播行为：

① PROPAGATION\_REQUIRED：如果当前没有事务，就创建一个新事务，如果当前存在事务，就加入该事务，该设置是最常用的设置。

② PROPAGATION\_SUPPORTS：支持当前事务，如果当前存在事务，就加入该事务，如果当前不存在事务，就以非事务执行。‘

③ PROPAGATION\_MANDATORY：支持当前事务，如果当前存在事务，就加入该事务，如果当前不存在事务，就抛出异常。

④ PROPAGATION\_REQUIRES\_NEW：创建新事务，无论当前存不存在事务，都创建新事务。

⑤ PROPAGATION\_NOT\_SUPPORTED：以非事务方式执行操作，如果当前存在事务，就把当前事务挂起。

⑥ PROPAGATION\_NEVER：以非事务方式执行，如果当前存在事务，则抛出异常。

⑦ PROPAGATION\_NESTED：如果当前存在事务，则在嵌套事务内执行。如果当前没有事务，则按 REQUIRED 属性执行。

（3）Spring中的隔离级别：

① ISOLATION\_DEFAULT：这是个 PlatfromTransactionManager 默认的隔离级别，使用数据库默认的事务隔离级别。

② ISOLATION\_READ\_UNCOMMITTED：读未提交，允许另外一个事务可以看到这个事务未提交的数据。

③ ISOLATION\_READ\_COMMITTED：读已提交，保证一个事务修改的数据提交后才能被另一事务读取，而且能看到该事务对已有记录的更新。

④ ISOLATION\_REPEATABLE\_READ：可重复读，保证一个事务修改的数据提交后才能被另一事务读取，但是不能看到该事务对已有记录的更新。

⑤ ISOLATION\_SERIALIZABLE：一个事务在执行的过程中完全看不到其他事务对数据库所做的更新