框架篇面试题-参考回答

面试官：Spring框架中的单例bean是线程安全的吗？

候选人：

嗯！

不是线程安全的，是这样的

当多用户同时请求一个服务时，容器会给每一个请求分配一个线程，这是多个线程会并发执行该请求对应的业务逻辑（成员方法），如果该处理逻辑中有对该单列状态的修改（体现为该单例的成员属性），则必须考虑线程同步问题。

Spring框架并没有对单例bean进行任何多线程的封装处理。关于单例bean的线程安全和并发问题需要开发者自行去搞定。

比如：我们通常在项目中使用的Spring bean都是不可可变的状态(比如Service类和DAO类)，所以在某种程度上说Spring的单例bean是线程安全的。

如果你的bean有多种状态的话（比如 View Model对象），就需要自行保证线程安全。最浅显的解决办法就是将多态bean的作用由“**singleton**”变更为“**prototype**”。

面试官：什么是AOP

候选人：

aop是面向切面编程，在spring中用于将那些与业务无关，但却对多个对象产生影响的公共行为和逻辑，抽取公共模块复用，降低耦合，一般比如可以做为公共日志保存，事务处理等

面试官：你们项目中有没有使用到AOP

候选人：

我们当时在后台管理系统中，就是使用aop来记录了系统的操作日志

主要思路是这样的，使用aop中的环绕通知+切点表达式，这个表达式就是要找到要记录日志的方法，然后通过环绕通知的参数获取请求方法的参数，比如类信息、方法信息、注解、请求方式等，获取到这些参数以后，保存到数据库

面试官：Spring中的事务是如何实现的

候选人：

spring实现的事务本质就是aop完成，对方法前后进行拦截，在执行方法之前开启事务，在执行完目标方法之后根据执行情况提交或者回滚事务。

面试官：Spring中事务失效的场景有哪些

候选人：

嗯！这个在项目中之前遇到过，我想想啊

第一个，如果方法上异常捕获处理，自己处理了异常，没有抛出，就会导致事务失效，所以一般处理了异常以后，别忘了跑出去就行了

第二个，如果方法抛出检查异常，如果报错也会导致事务失效，最后在spring事务的注解上，就是@Transactional上配置rollbackFor属性为Exception，这样别管是什么异常，都会回滚事务

第三，我之前还遇到过一个，如果方法上不是public修饰的，也会导致事务失效

嗯，就能想起来那么多

面试官：Spring的bean的生命周期

候选人：

嗯！，这个步骤还是挺多的，我之前看过一些源码，它大概流程是这样的

首先会通过一个非常重要的类，叫做BeanDefinition获取bean的定义信息，这里面就封装了bean的所有信息，比如，类的全路径，是否是延迟加载，是否是单例等等这些信息

在创建bean的时候，第一步是调用构造函数实例化bean

第二步是bean的依赖注入，比如一些set方法注入，像平时开发用的@Autowire都是这一步完成

第三步是处理Aware接口，如果某一个bean实现了Aware接口就会重写方法执行

第四步是bean的后置处理器BeanPostProcessor，这个是前置处理器

第五步是初始化方法，比如实现了接口InitializingBean或者自定义了方法init-method标签或@PostContruct

第六步是执行了bean的后置处理器BeanPostProcessor，主要是对bean进行增强，有可能在这里产生代理对象

最后一步是销毁bean

面试官：Spring中的循环引用

候选人：

嗯，好的，我来解释一下

循环依赖：循环依赖其实就是循环引用,也就是两个或两个以上的bean互相持有对方,最终形成闭环。比如A依赖于B,B依赖于A

循环依赖在spring中是允许存在，spring框架依据三级缓存已经解决了大部分的循环依赖

①一级缓存：单例池，缓存已经经历了完整的生命周期，已经初始化完成的bean对象

②二级缓存：缓存早期的bean对象（生命周期还没走完）

③三级缓存：缓存的是ObjectFactory，表示对象工厂，用来创建某个对象的

面试官：那具体解决流程清楚吗？

候选人：

第一，先实例A对象，同时会创建ObjectFactory对象存入三级缓存singletonFactories

第二，A在初始化的时候需要B对象，这个走B的创建的逻辑

第三，B实例化完成，也会创建ObjectFactory对象存入三级缓存singletonFactories

第四，B需要注入A，通过三级缓存中获取ObjectFactory来生成一个A的对象同时存入二级缓存，这个是有两种情况，一个是可能是A的普通对象，另外一个是A的代理对象，都可以让ObjectFactory来生产对应的对象，这也是三级缓存的关键

第五，B通过从通过二级缓存earlySingletonObjects 获得到A的对象后可以正常注入，B创建成功，存入一级缓存singletonObjects

第六，回到A对象初始化，因为B对象已经创建完成，则可以直接注入B，A创建成功存入一次缓存singletonObjects

第七，二级缓存中的临时对象A清除

面试官：构造方法出现了循环依赖怎么解决？

候选人：

由于bean的生命周期中构造函数是第一个执行的，spring框架并不能解决构造函数的的依赖注入，可以使用@Lazy懒加载，什么时候需要对象再进行bean对象的创建

面试官：SpringMVC的执行流程知道嘛

候选人：

嗯，这个知道的，它分了好多步骤

1、用户发送出请求到前端控制器DispatcherServlet，这是一个调度中心

2、DispatcherServlet收到请求调用HandlerMapping（处理器映射器）。

3、HandlerMapping找到具体的处理器(可查找xml配置或注解配置)，生成处理器对象及处理器拦截器(如果有)，再一起返回给DispatcherServlet。

4、DispatcherServlet调用HandlerAdapter（处理器适配器）。

5、HandlerAdapter经过适配调用具体的处理器（Handler/Controller）。

6、Controller执行完成返回ModelAndView对象。

7、HandlerAdapter将Controller执行结果ModelAndView返回给DispatcherServlet。

8、DispatcherServlet将ModelAndView传给ViewReslover（视图解析器）。

9、ViewReslover解析后返回具体View（视图）。

10、DispatcherServlet根据View进行渲染视图（即将模型数据填充至视图中）。

11、DispatcherServlet响应用户。

当然现在的开发，基本都是前后端分离的开发的，并没有视图这些，一般都是handler中使用Response直接结果返回

面试官：Springboot自动配置原理

候选人：

嗯，好的，它是这样的。

在Spring Boot项目中的引导类上有一个注解@SpringBootApplication，这个注解是对三个注解进行了封装，分别是：

- @SpringBootConfiguration
- @EnableAutoConfiguration
- @ComponentScan

其中`@EnableAutoConfiguration`是实现自动化配置的核心注解。

该注解通过`@Import`注解导入对应的配置选择器。关键的是内部就是读取了该项目和该项目引用的Jar包的的classpath路径下**META-INF/spring.factories**文件中的所配置的类的全类名。

在这些配置类中所定义的Bean会根据条件注解所**指定的条件来决定**是否需要将其导入到Spring容器中。

一般条件判断会有像`@ConditionalOnClass`这样的注解，判断是否有对应的class文件，如果有则加载该类，把这个配置类的所有的Bean放入spring容器中使用。

面试官：Spring 的常见注解有哪些？

候选人：

嗯，这个就很多了

第一类是：声明bean，有@Component、@Service、@Repository、@Controller

第二类是：依赖注入相关的，有@Autowired、@Qualifier、@Resourse

第三类是：设置作用域 @Scope

第四类是：spring配置相关的，比如@Configuration，@ComponentScan 和@Bean

第五类是：跟aop相关做增强的注解 @Aspect，@Before，@After，@Around，@Pointcut

面试官：SpringMVC常见的注解有哪些？

候选人：

嗯，这个也很多的

有@RequestMapping：用于映射请求路径；

@RequestBody：注解实现接收http请求的json数据，将json转换为java对象；

@RequestParam：指定请求参数的名称；

@PathViriable：从请求路径下中获取请求参数(/user/{id})，传递给方法的形式参数；@ResponseBody：注解实现将controller方法返回对象转化为json对象响应给客户端。@RequestHeader：获取指定的请求头数据，还有像@PostMapping、@GetMapping这些。

面试官：Springboot常见注解有哪些？

候选人：

嗯~~

Spring Boot的核心注解是@SpringBootApplication , 他由几个注解组成 :

- @SpringBootConfiguration： 组合了- @Configuration注解，实现配置文件的功能；
- @EnableAutoConfiguration：打开自动配置的功能，也可以关闭某个自动配置的选项
- @ComponentScan：Spring组件扫描

面试官：MyBatis执行流程

候选人：

好，这个知道的，不过步骤也很多

①读取MyBatis配置文件：mybatis-config.xml加载运行环境和映射文件

②构造会话工厂SqlSessionFactory，一个项目只需要一个，单例的，一般由spring进行管理

③会话工厂创建SqlSession对象，这里面就含了执行SQL语句的所有方法

④操作数据库的接口，Executor执行器，同时负责查询缓存的维护

⑤Executor接口的执行方法中有一个MappedStatement类型的参数，封装了映射信息

⑥输入参数映射

⑦输出结果映射

面试官：Mybatis是否支持延迟加载？

候选人：

是支持的~

延迟加载的意思是：就是在需要用到数据时才进行加载，不需要用到数据时就不加载数据。

Mybatis支持一对一关联对象和一对多关联集合对象的延迟加载

在Mybatis配置文件中，可以配置是否启用延迟加载
lazyLoadingEnabled=true|false，默认是关闭的

面试官：延迟加载的底层原理知道吗？

候选人：

嗯，我想想啊

延迟加载在底层主要使用的CGLIB动态代理完成的

第一是，使用CGLIB创建目标对象的代理对象，这里的目标对象就是开启了延迟加载的mapper

第二个是当调用目标方法时，进入拦截器invoke方法，发现目标方法是null值，再执行sql查询

第三个是获取数据以后，调用set方法设置属性值，再继续查询目标方法，就有值了

面试官：Mybatis的一级、二级缓存用过吗？

候选人：

嗯~~，用过的~

mybatis的一级缓存: 基于 PerpetualCache 的 HashMap 本地缓存，其存储作用域为 Session，当Session进行flush或close之后，该Session中的所有Cache就将清空，默认打开一级缓存

关于二级缓存需要单独开启

二级缓存是基于namespace和mapper的作用域起作用的，不是依赖于SQL session，默认也是采用 PerpetualCache，HashMap 存储。

如果想要开启二级缓存需要在全局配置文件和映射文件中开启配置才行。

面试官：Mybatis的二级缓存什么时候会清理缓存中的数据

候选人：

嗯！！

当某一个作用域(一级缓存 Session/二级缓存Namespaces)的进行了新增、修改、删除操作后，默认该作用域下所有 select 中的缓存将被 clear。