链滴

面试 |Spring 的常见面试题

作者：qq692310342

原文链接：https://ld246.com/article/1565622754084

来源网站：链滴

什么是 Spring 框架?

对于JAVA开发者而言，这是一个很好的时代，相对于以前的EJB，Spring提供了更加轻量级和简单的程模型，它增强了简单老式的Java对象的功能；

Spring是一个开源的，同时也是一个简化开发的框架；

我们一般说 Spring 框架指的都是 Spring Framework，它是很多模块的集合，使用这些模块可以很便地协助我们进行开发。这些模块是：核心容器、数据访问/集成、Web、AOP（面向切面编程）、具、消息和测试模块。比如：Core Container 中的 Core 组件是Spring 所有组件的核心，Beans 组和 Context 组件是实现IOC和依赖注入的基础，AOP组件用来实现面向切面编程。

Spring 官网列出的 Spring 的 6 个特征:

- **核心技术**： 依赖注入(DI)，AOP，事件(events)，资源，i18n，验证，数据绑定，类型转换，SpEL。
- **测试**： 模拟对象，TestContext框架，Spring MVC 测试，WebTestClient。
- **数据访问**： 事务，DAO支持，JDBC，ORM，编组XML。
- **Web支持**: Spring MVC和Spring WebFlux Web框架。
- **集成**： 远程处理，JMS，JCA，JMX，电子邮件，任务，调度，缓存。
- **语言**： Kotlin，Groovy，动态语言。

Spring是如何实现简化开发的?

实现简化开发的方式:

1. 基于POJO的轻量级和最小侵入式编程

传统框架需要强迫某个类要去继承或者实现某个接口从而导致的应用与应用的绑定，无法松耦合，而Spring提供的是一个非入侵式的编程方式，让类仅仅实现一个注解就能实现POJO的在Spring应用和非Sping应用中使用。

2. 通过依赖注入（DI）和面向接口实现的松耦合

在传统的开发模式中，我们需要通过实现一个甚至多个的实现类来模拟出测试环境，而通常这样就会试环境耦合度过高，从而不好判断出是哪一个地方出现的问题；而Spring给我们提供的Dependence Ifection机制就能够让我们专注于当前类的测试，而不用太过关注与其他有相关依赖作用的类，这就是耦合的体现；

而耦合也具有两面性，一定程度的耦合又是必须的，完全没有耦合的代码什么都做不了；

通过DI，对象的依赖关系将会通过系统中负责协调各对象的注入的容器进行统一管理；

接口就是标准规范，一种规范约束，有了**标准去遵守就容易扩展**！我们**只需要面向标准编程**，而不用对具体的实现类！面向接口编程好处：（1）抽象；（2）高内聚，低耦合，降低开发成本/维护成本

3. 基于切面和惯例进行声明式编程风格

面向切边编程（aspect oriented programming）允许我们把遍布应用的功能性代码得到分离开来，成可以复用的组件。十分典型的一个例子就是利用AOP来实现的事务控制。

组件会因为那些与自身的核心业务无关的代码变得十分复杂，所以我们希望更多的关注业务本身，而是那些事务控制的组件，这也是一种简化开发的实现了。

而基于声明式的编程风格，可以实现代码的复用，也是一个简化开发的体现。

4. 通过切面和模块来减少样板式代码

Spring全家桶也提供了许多的模板来实现简化开发，例如我们在写一段jdbc连接数据库进行CRUD的候，就需要不断的重复写一些冗余的代码，而Spring就提供了一个JdbcTemplate来简化我们的对于据库的一个单表操作的开发，类似的还有spring提供的spring Data JPA、redisTemplate等等便捷的具。

列举一些重要的Spring模块?

下图对应的是 Spring4.x 版本。目前最新的5.x版本中 Web 模块的 Portlet 组件已经被废弃掉，同时加了用于异步响应式处理的 WebFlux 组件。nullSpring主要模块

Spring Core： 基础,可以说 Spring 其他所有的功能都需要依赖于该类库。主要提供 IOC 依赖注入功。

Spring Aspects ： 该模块为与AspectJ的集成提供支持。

Spring AOP ： 提供了面向方面的编程实现。

Spring JDBC : Java数据库连接。

Spring JMS ： Java消息服务。

Spring ORM : 用于支持Hibernate等ORM工具。

Spring Web : 为创建Web应用程序提供支持。

Spring Test : 提供了对 JUnit 和 TestNG 测试的支持。

谈谈自己对于 Spring IoC 和 AOP 的理解

IoC

IoC（Inverse of Control:控制反转）是一种设计思想，就是 将原本在程序中手动创建对象的控制权交由Spring框架来管理。 IoC 在其他语言中也有应用，并非 Spirng 特有。 IoC 容器是 Spring 用来现 IoC 的载体， IoC 容器实际上就是个Map（key，value）,Map 中存放的是各种对象。

将对象之间的相互依赖关系交给 IOC 容器来管理，并由 IOC 容器完成对象的注入。这样可以很大程

上简化应用的开发，把应用从复杂的依赖关系中解放出来。 IOC 容器就像是一个工厂一样，当我们要创建一个对象的时候，只需要配置好配置文件/注解即可，完全不用考虑对象是如何被创建出来的。在实际项目中一个 Service 类可能有几百甚至上千个类作为它的底层，假如我们需要实例化这个 Service，你可能要每次都要搞清这个 Service 所有底层类的构造函数，这可能会把人逼疯。如果利用 IOC 话，你只需要配置好，然后在需要的地方引用就行了，这大大增加了项目的可维护性且降低了开发难。

Spring 时代我们一般通过 XML 文件来配置 Bean，后来开发人员觉得 XML 文件来配置不太好，于是 SpringBoot 注解配置就慢慢开始流行起来。

推荐阅读：https://www.zhihu.com/question/23277575/answer/169698662

Spring IOC的初始化过程：

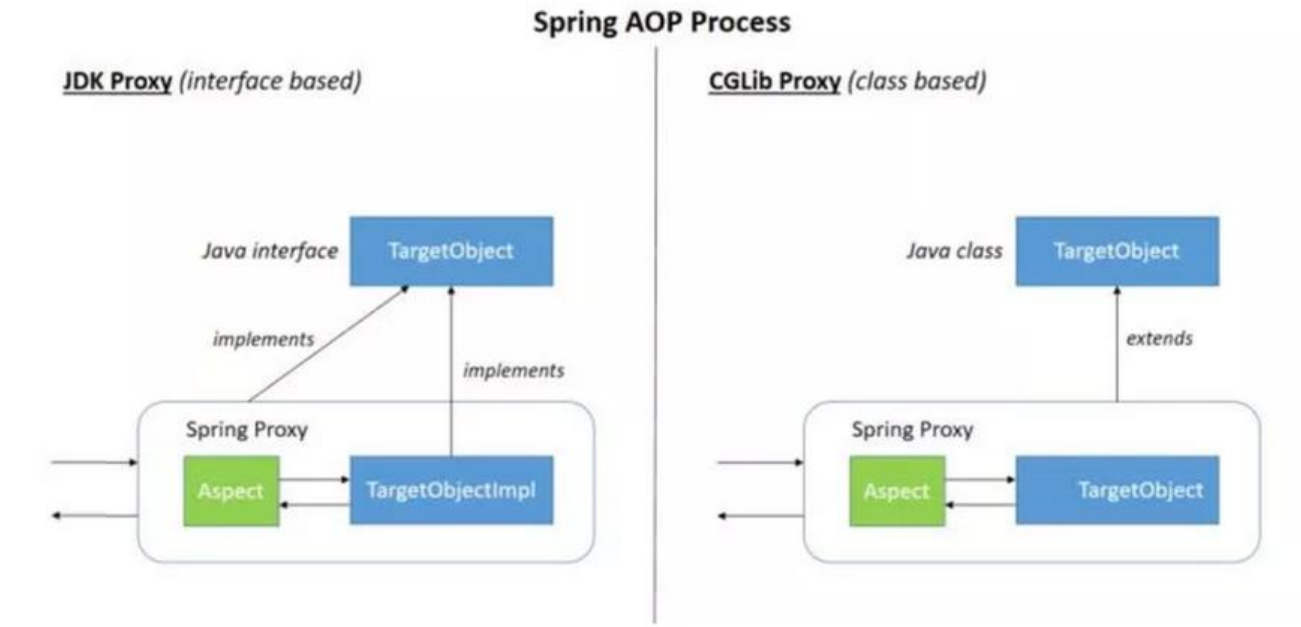
IOC源码阅读

https://javadoop.com/post/spring-ioc

AOP

AOP(Aspect-Oriented Programming:面向切面编程)能够将那些与业务无关，却为业务模块所共同用的逻辑或责任（例如事务处理、日志管理、权限控制等）封装起来，便于减少系统的重复代码，降模块间的耦合度，并有利于未来的可拓展性和可维护性。

Spring AOP就是基于动态代理的，如果要代理的对象，实现了某个接口，那么Spring AOP会使用JDK Proxy，去创建代理对象，而对于没有实现接口的对象，就无法使用 JDK Proxy 去进行代理了，这时Spring AOP会使用Cglib ，这时候Spring AOP会使用 Cglib 生成一个被代理对象的子类来作为代理如下图所示：



当然你也可以使用 AspectJ ,Spring AOP 已经集成了AspectJ ， AspectJ 应该算的上是 Java 生态系中最完整的 AOP 框架了。

使用 AOP 之后我们可以把一些通用功能抽象出来，在需要用到的地方直接使用即可，这样大大简化

代码量。我们需要增加新功能时也方便，这样也提高了系统扩展性。日志功能、事务管理等等场景都到了 AOP 。

Spring AOP 和 AspectJ AOP 有什么区别?

Spring AOP 属于运行时增强，而 AspectJ 是编译时增强。 Spring AOP 基于代理(Proxying)，而 AsectJ 基于字节码操作(Bytecode Manipulation)。

Spring AOP 已经集成了 AspectJ ，AspectJ 应该算的上是 Java 生态系统中最完整的 AOP 框架了。spectJ 相比于 Spring AOP 功能更加强大，但是 Spring AOP 相对来说更简单，

如果我们的切面比较少，那么两者性能差异不大。但是，当切面太多的话，最好选择 AspectJ ，它比pring AOP 快很多。

Spring 中的 bean 的作用域有哪些?

singleton : 唯一 bean 实例，Spring 中的 bean 默认都是单例的。

prototype : 每次请求都会创建一个新的 bean 实例。

request : 每一次HTTP请求都会产生一个新的bean，该bean仅在当前HTTP request内有效。

session : 每一次HTTP请求都会产生一个新的 bean，该bean仅在当前 HTTP session 内有效。

global-session：全局session作用域，仅仅在基于portlet的web应用中才有意义，Spring5已经没有。Portlet是能够生成语义代码(例如：HTML)片段的小型Java Web插件。它们基于portlet容器，可像servlet一样处理HTTP请求。但是，与 servlet 不同，每个 portlet 都有不同的会话

Spring 中的单例 bean 的线程安全问题了解吗?

大部分时候我们并没有在系统中使用多线程，所以很少有人会关注这个问题。单例 bean 存在线程问，主要是因为当多个线程操作同一个对象的时候，对这个对象的非静态成员变量的写操作会存在线程全问题。

常见的有两种解决办法：

在Bean对象中尽量避免定义可变的成员变量（不太现实）。

在类中定义一个ThreadLocal成员变量，将需要的可变成员变量保存在 ThreadLocal 中（推荐的一种式）。

Spring 中的 bean 生命周期?

这部分网上有很多文章都讲到了，下面的内容整理自：https://yemengying.com/2016/07/14/sprin-bean-life-cycle/ ，除了这篇文章，再推荐一篇很不错的文章： https://www.cnblogs.com/zrtqsk//3735273.html 。

Bean 容器找到配置文件中 Spring Bean 的定义。

Bean 容器利用 Java Reflection API 创建一个Bean的实例。

如果涉及到一些属性值 利用 set()方法设置一些属性值。

如果 Bean 实现了 BeanNameAware 接口，调用 setBeanName()方法，传入Bean的名字。

如果 Bean 实现了 BeanClassLoaderAware 接口，调用 setBeanClassLoader()方法，传入 ClassLoa

er对象的实例。

如果Bean实现了 BeanFactoryAware 接口，调用 setBeanClassLoader()方法，传入 ClassLoade r 象的实例。

与上面的类似，如果实现了其他 *.Aware接口，就调用相应的方法。

如果有和加载这个 Bean 的 Spring 容器相关的 BeanPostProcessor 对象，执行postProcessBeforeIn tialization() 方法

如果Bean实现了InitializingBean接口，执行afterPropertiesSet()方法。

如果 Bean 在配置文件中的定义包含 init-method 属性，执行指定的方法。

如果有和加载这个 Bean的 Spring 容器相关的 BeanPostProcessor 对象，执行postProcessAfterInit alization() 方法

当要销毁 Bean 的时候，如果 Bean 实现了 DisposableBean 接口，执行 destroy() 方法。

当要销毁 Bean 的时候，如果 Bean 在配置文件中的定义包含 destroy-method 属性，执行指定的方 。

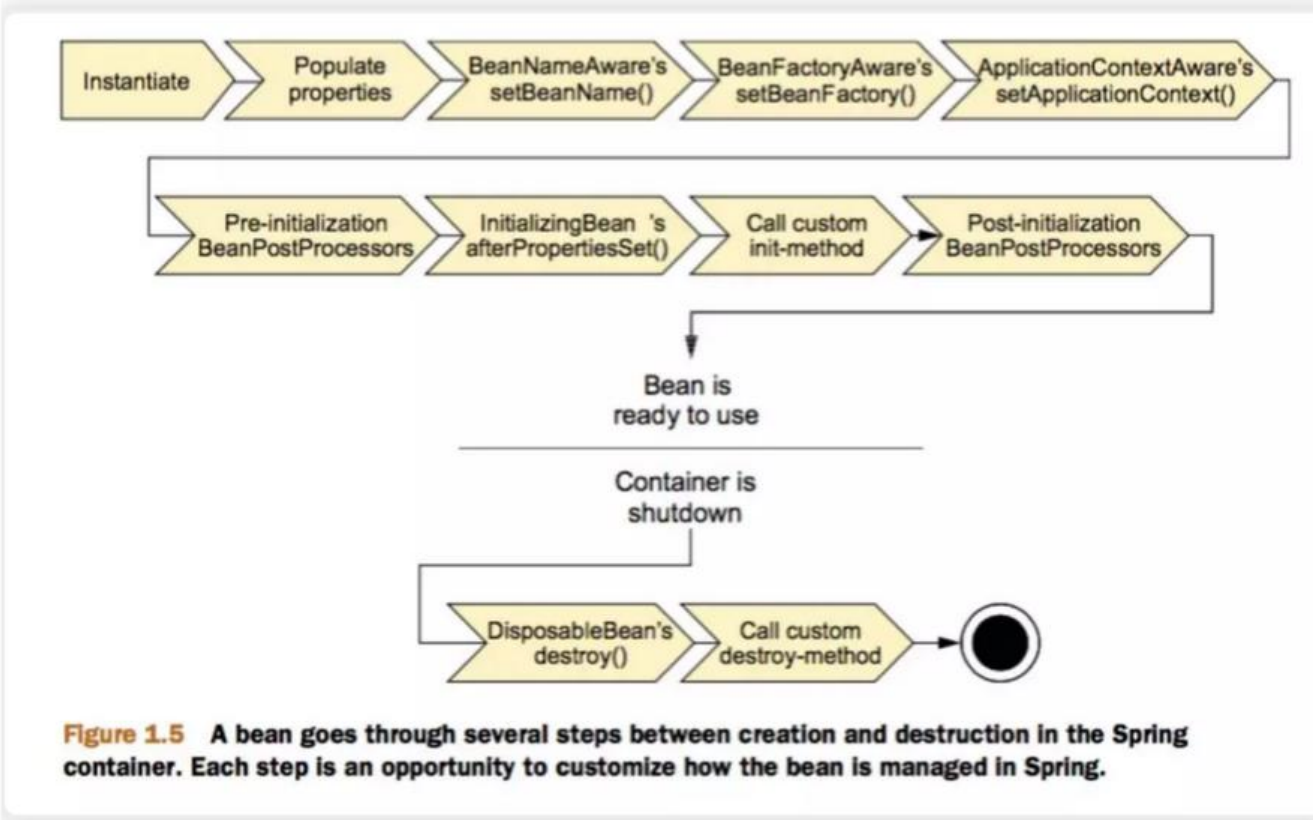
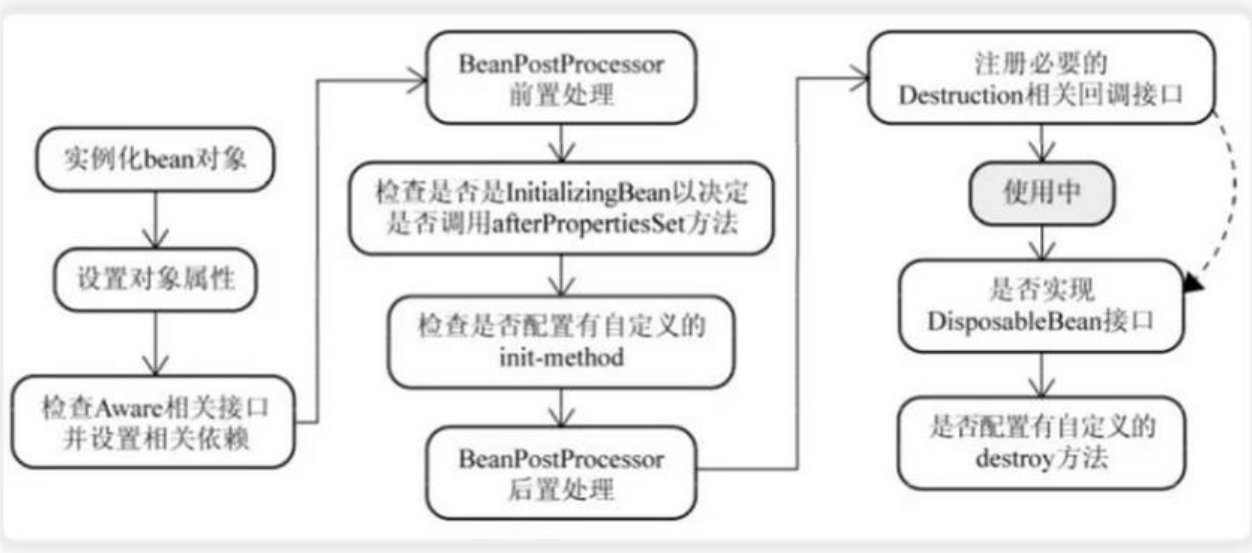


Figure 1.5 A bean goes through several steps between creation and destruction in the Spring container. Each step is an opportunity to customize how the bean is managed in Spring.

与之比较类似的中文版本:



说说自己对于 Spring MVC 了解?

谈到这个问题，我们不得不提提之前 Model1 和 Model2 这两个没有 Spring MVC 的时代。

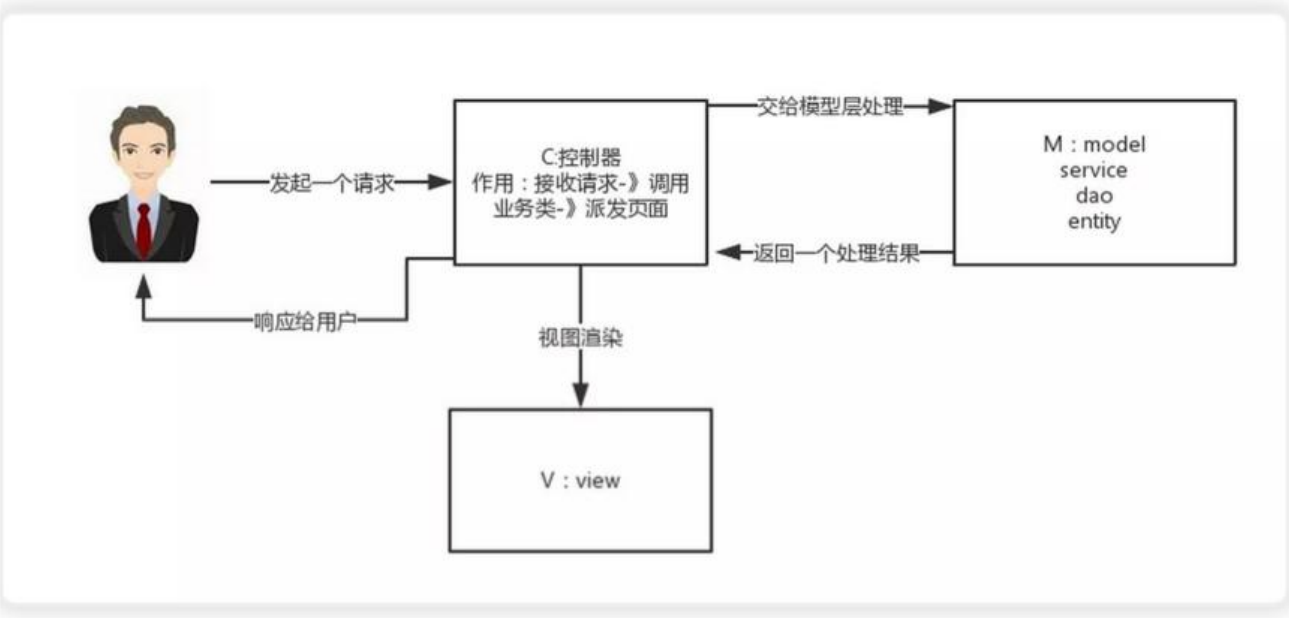
1. Model1 时代 : 很多学 Java 后端比较晚的朋友可能并没有接触过 Model1 模式下的 JavaWeb 应开发。在 Model1 模式下，整个 Web 应用几乎全部用 JSP 页面组成，只用少量的 JavaBean 来处理据库连接、访问等操作。这个模式下 JSP 即是控制层又是表现层。显而易见，这种模式存在很多问题比如①将控制逻辑和表现逻辑混杂在一起，导致代码重用率极低；②前端和后端相互依赖，难以进行试并且开发效率极低;

2. Model2 时代： 学过 Servlet 并做过相关 Demo 的朋友应该了解“Java Bean(Model)+ JSP（View）+Servlet（Controller）”这种开发模式,这就是早期的 JavaWeb MVC 开发模式。Model:系统涉的数据，也就是 dao 和 bean。View：展示模型中的数据，只是用来展示。Controller：处理用户请都发送给 ，返回数据给 JSP 并展示给用户。

Model2 模式下还存在很多问题，Model2的抽象和封装程度还远远不够，使用Model2进行开发时不避免地会重复造轮子，这就大大降低了程序的可维护性和复用性。于是很多JavaWeb开发相关的 MVC 框架营运而生比如Struts2，但是 Struts2 比较笨重。随着 Spring 轻量级开发框架的流行，Spring 态圈出现了 Spring MVC 框架， Spring MVC 是当前最优秀的 MVC 框架。相比于 Struts2， Spring MVC 使用更加简单和方便，开发效率更高，并且 Spring MVC 运行速度更快。

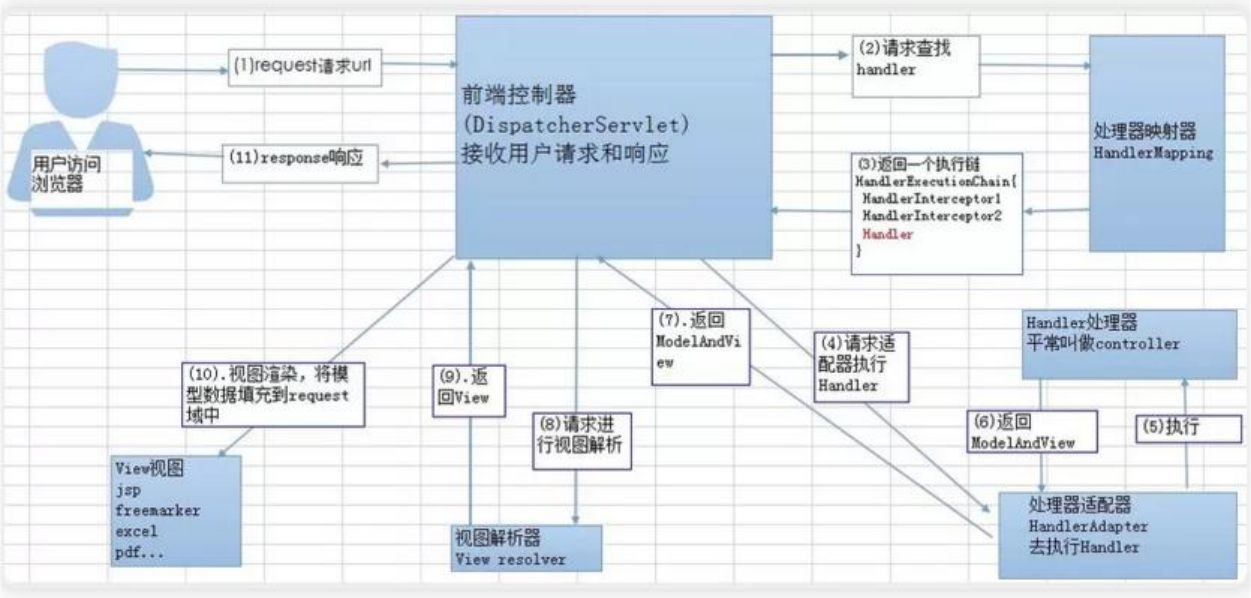
MVC 是一种设计模式,Spring MVC 是一款很优秀的 MVC 框架。Spring MVC 可以帮助我们进行更洁的Web层的开发，并且它天生与 Spring 框架集成。Spring MVC 下我们一般把后端项目分为 Servie层（处理业务）、Dao层（数据库操作）、Entity层（实体类）、Controller层(控制层，返回数据给台页面)。

Spring MVC 的简单原理图如下:



SpringMVC 工作原理了解吗?

原理如下图所示：



上图的一个笔误的小问题：Spring MVC 的入口函数也就是前端控制器 DispatcherServlet 的作用是收请求，响应结果。

流程说明（重要）：

客户端（浏览器）发送请求，直接请求到 DispatcherServlet。

DispatcherServlet 根据请求信息调用 HandlerMapping，解析请求对应的 Handler。

解析到对应的 Handler（也就是我们平常说的 Controller 控制器）后，开始由 HandlerAdapter 适器处理。

HandlerAdapter 会根据 Handler来调用真正的处理器开处理请求，并处理相应的业务逻辑。

处理器处理完业务后，会返回一个 ModelAndView 对象，Model 是返回的数据对象，View 是个逻

上的 View。

ViewResolver 会根据逻辑 View 查找实际的 View。

DispaterServlet 把返回的 Model 传给 View（视图渲染）。

把 View 返回给请求者（浏览器）

Spring 框架中用到了哪些设计模式?

工厂设计模式 : Spring使用工厂模式通过 BeanFactory、ApplicationContext 创建 bean 对象。

代理设计模式 : Spring AOP 功能的实现。

单例设计模式 : Spring 中的 Bean 默认都是单例的。

模板方法模式 : Spring 中 jdbcTemplate、hibernateTemplate 等以 Template 结尾的对数据库操的类，它们就使用到了模板模式。

包装器设计模式 : 我们的项目需要连接多个数据库，而且不同的客户在每次访问中根据需要会去访问同的数据库。这种模式让我们可以根据客户的需求能够动态切换不同的数据源。

观察者模式: Spring 事件驱动模型就是观察者模式很经典的一个应用。

适配器模式 :Spring AOP 的增强或通知(Advice)使用到了适配器模式、spring MVC 中也是用到了适器模式适配Controller。

……

@Component 和 @Bean 的区别是什么?

作用对象不同: @Component 注解作用于类，而@Bean注解作用于方法。

@Component通常是通过类路径扫描来自动侦测以及自动装配到Spring容器中（我们可以使用 @CmponentScan 注解定义要扫描的路径从中找出标识了需要装配的类自动装配到 Spring 的 bean 容中）。@Bean 注解通常是我们在标有该注解的方法中定义产生这个 bean,@Bean告诉了Spring这是个类的示例，当我需要用它的时候还给我。

@Bean 注解比 Component 注解的自定义性更强，而且很多地方我们只能通过 @Bean 注解来注册ban。比如当我们引用第三方库中的类需要装配到 Spring容器时，则只能通过 @Bean来实现。

@Bean注解使用示例：

```
@Configuration
public class AppConfig {
    @Bean
    public TransferService transferService() {
        return new TransferServiceImpl();
    }

}
```

上面的代码相当于下面的 xml 配置

```
<beans>
    <bean id="transferService" class="com.acme.TransferServiceImpl"/>
</beans>
```

下面这个例子是通过 @Component 无法实现的。

```
@Bean
public OneService getService(status) {
    case (status)  {
        when 1:
                return new serviceImpl1();
        when 2:
                return new serviceImpl2();
        when 3:
                return new serviceImpl3();
    }
}
```

将一个类声明为Spring的 bean 的注解有哪些?

我们一般使用 @Autowired 注解自动装配 bean，要想把类标识成可用于 @Autowired注解自动装的 bean 的类,采用以下注解可实现：

@Component ： 通用的注解，可标注任意类为 Spring 组件。如果一个Bean不知道属于拿个层，可使用@Component 注解标注。

@Repository : 对应持久层即 Dao 层，主要用于数据库相关操作。

@Service : 对应服务层，主要涉及一些复杂的逻辑，需要用到 Dao层。

@Controller : 对应 Spring MVC 控制层，主要用户接受用户请求并调用 Service 层返回数据给前端面。

Spring 管理事务的方式有几种?

编程式事务，在代码中硬编码。(不推荐使用)

声明式事务，在配置文件中配置（推荐使用）

声明式事务又分为两种：

基于XML的声明式事务

基于注解的声明式事务

Spring 事务中的隔离级别有哪几种?

TransactionDefinition 接口中定义了五个表示隔离级别的常量：

TransactionDefinition.ISOLATION_DEFAULT: 使用后端数据库默认的隔离级别，Mysql 默认采用的EPEATABLE_READ隔离级别 Oracle 默认采用的 READ_COMMITTED隔离级别.

TransactionDefinition.ISOLATION_READ_UNCOMMITTED: 最低的隔离级别，允许读取尚未提交数据变更，可能会导致脏读、幻读或不可重复读

TransactionDefinition.ISOLATION_READ_COMMITTED: 允许读取并发事务已经提交的数据，可以止脏读，但是幻读或不可重复读仍有可能发生

TransactionDefinition.ISOLATION_REPEATABLE_READ: 对同一字段的多次读取结果都是一致的，非数据是被本身事务自己所修改，可以阻止脏读和不可重复读，但幻读仍有可能发生。

TransactionDefinition.ISOLATION_SERIALIZABLE: 最高的隔离级别，完全服从ACID的隔离级别。有的事务依次逐个执行，这样事务之间就完全不可能产生干扰，也就是说，该级别可以防止脏读、不重复读以及幻读。但是这将严重影响程序的性能。通常情况下也不会用到该级别。

Spring 事务中哪几种事务传播行为?

支持当前事务的情况：

TransactionDefinition.PROPAGATION_REQUIRED： 如果当前存在事务，则加入该事务；如果当没有事务，则创建一个新的事务。

TransactionDefinition.PROPAGATION_SUPPORTS： 如果当前存在事务，则加入该事务；如果当没有事务，则以非事务的方式继续运行。

TransactionDefinition.PROPAGATION_MANDATORY： 如果当前存在事务，则加入该事务；如果前没有事务，则抛出异常。（mandatory：强制性）

不支持当前事务的情况：

TransactionDefinition.PROPAGATION_REQUIRES_NEW： 创建一个新的事务，如果当前存在事务则把当前事务挂起。

TransactionDefinition.PROPAGATION_NOT_SUPPORTED： 以非事务方式运行，如果当前存在事，则把当前事务挂起。

TransactionDefinition.PROPAGATION_NEVER： 以非事务方式运行，如果当前存在事务，则抛出常。

其他情况：

TransactionDefinition.PROPAGATION_NESTED： 如果当前存在事务，则创建一个事务作为当前务的嵌套事务来运行；如果当前没有事务，则该取值等价于TransactionDefinition.PROPAGATION_EQUIRED。

END

2019年8月13日22:54:02

以上部分内容转载自公众号JavaGuide，作者SnialClimb