第三版：SpringBoot 25 道

SpringBoot 最大的优势是什么呢?

SpringBoot 的最大的优势是“约定优于配置“。“约定优于配置“是一种软件设计范式，开发人员按照约定的方式来进行编程，可以减少软件开发人员需做决定的数量，获得简单的好处，而又不失灵活性。

SpringBoot 中 “约定优于配置“的具体产品体现在哪里。

SpringBoot Starter、SpringBoot Jpa 都是“约定优于配置“的一种体现。都是通过“约定优于配置“的设计思路来设计的，SpringBoot Starter 在启动的过程中会根据约定的信息对资源进行初始化；SpringBoot Jpa 通过约定的方式来自动生成 Sql ，避免大量无效代码编写。具体详细可以参考：SpringBoot 为什么这么火?

SpringBoot Starter 的工作原理是什么?

SpringBoot 在启动的时候会干这几件事情：

1、 SpringBoot 在启动时会去依赖的 Starter 包中寻找 resources/META-INF/spring.factories 文件，然后根据文件中配置的 Jar 包去扫描项目所依赖的 Jar 包。

2、 根据 spring.factories 配置加载 AutoConfigure 类

3、 根据 @Conditional 注解的条件，进行自动配置并将 Bean 注入 Spring Context

总结一下，其实就是 SpringBoot 在启动的时候，按照约定去读取 SpringBoot Starter 的配置信息，再根据配置信息对资源进行初始化，并注入到 Spring 容器中。这样 SpringBoot 启动完毕后，就已经准备好了一切资源，使用过程中直接注入对应 Bean 资源即可。

这只是简单的三连环问答，不知道有多少同学能够完整的回答出来。

其实 SpringBoot 中有很多的技术点可以挖掘，今天给大家整理了十个高频 SpringBoot 面试题，希望可以在后期的面试中帮助到大家。

SpringBoot 的自动配置是如何实现的?

SpringBoot 项目的启动注解是：@SpringBootApplication，其实它就是由下面三个注解组成的：

1、 @Configuration

2、 @ComponentScan

3、 @EnableAutoConfiguration

其中 @EnableAutoConfiguration 是实现自动配置的入口，该注解又通过 @Import 注解导入了 AutoConfigurationImportSelector，在该类中加载 META-INF/spring.factories 的配置信息。然后筛选出以 EnableAutoConfiguration 为 key 的数据，加载到 IOC 容器中，实现自动配置功能！

什么是嵌入式服务器？我们为什么要使用嵌入式服务器呢?

思考一下在你的虚拟机上部署应用程序需要些什么。

第一步：安装 Java

第二部：安装 Web 或者是应用程序的服务器（Tomat/Wbesphere/Weblogic 等等）

第三部：部署应用程序 war 包

如果我们想简化这些步骤，应该如何做呢？

让我们来思考如何使服务器成为应用程序的一部分？

你只需要一个安装了 Java 的虚拟机，就可以直接在上面部署应用程序了，

是不是很爽？

这个想法是嵌入式服务器的起源。

当我们创建一个可以部署的应用程序的时候，我们将会把服务器（例如，tomcat）嵌入到可部署的服务器中。

例如，对于一个 SpringBoot 应用程序来说，你可以生成一个包含 Embedded Tomcat 的应用程序 jar。你就可以像运行正常 Java 应用程序一样来运行 web 应用程序了。

嵌入式服务器就是我们的可执行单元包含服务器的二进制文件(例如，tomcat.jar)。

微服务同时调用多个接口，怎么支持事务的啊？

支持分布式事务，可以使用 SpringBoot 集成 Aatomikos 来解决，但是我一般不建议这样使用，因为使用分布式事务会增加请求的响应时间，影响系统的 TPS。一般在实际工作中，会利用消息的补偿机制来处理分布式的事务。

shiro 和 oauth 还有 cas 他们之间的关系是什么？问下您公司权限是如何设计，还有就是这几个概念的区别。

cas 和 oauth 是一个解决单点登录的组件，shiro 主要是负责权限安全方面的工作，所以功能点不一致。但往往需要单点登陆和权限控制一起来使用，所以就有 cas+shiro 或者 oauth+shiro 这样的组合。

token 一般是客户端登录后服务端生成的令牌，每次访问服务端会进行校验，一般保存到内存即可，也可以放到其他介质；Redis 可以做 Session 共享，如果前端 web 服务器有几台负载，但是需要保持用户登录的状态，这场景使用比较常见。

我们公司使用 oauth+shiro 这样的方式来做后台权限的管理，oauth 负责多后台统一登录认证，shiro 负责给登录用户赋予不同的访问权限。

各服务之间通信，对 Restful 和 Rpc 这 2 种方式如何做选择?

在传统的 SOA 治理中，使用 rpc 的居多；Spring Cloud 默认使用 restful 进行服务之间的通讯。rpc 通讯效率会比 restful 要高一些，但是对于大多数公司来讲，这点效率影响甚微。我建议使用 restful 这种方式，易于在不同语言实现的服务之间通讯。

怎么设计无状态服务?

对于无状态服务，首先说一下什么是状态：如果一个数据需要被多个服务共享，才能完成一笔交易，那么这 1、个数据被称为状态。进而依赖这个“状态”数据的服务被称为有状态服务，反之称为无状态服务。

2、 那么这个无状态服务原则并不是说在微服务架构里就不允许存在状态，表达的真实意思是要把有状态的业务服务改变为无状态的计算类服务，那么状态数据也就相应的迁移到对应的“有状态数据服务”中。

3、 场景说明：例如我们以前在本地内存中建立的数据缓存、Session 缓存，到现在的微服务架构中就应该把这些数据迁移到分布式缓存中存储，让业务服务变成一个无状态的计算节点。迁移后，就可以做到按需动态伸缩，微服务应用在运行时动态增删节点，就不再需要考虑缓存数据如何同步的问题。

Spring Cache 三种常用的缓存注解和意义?

1、 @Cacheable，用来声明方法是可缓存，将结果存储到缓存中以便后续使用相同参数调用时不需执行实际的方法，直接从缓存中取值。

2、 @CachePut，使用 @CachePut 标注的方法在执行前，不会去检查缓存中是否存在之前执行过的结果，而是每次都会执行该方法，并将执行结果以键值对的形式存入指定的缓存中。

3、 @CacheEvict，是用来标注在需要清除缓存元素的方法或类上的，当标记在一个类上时表示其中所有的方法的执行都会触发缓存的清除操作。

SpringBoot 如何设置支持跨域请求?

现代浏览器出于安全的考虑， HTTP 请求时必须遵守同源策略，否则就是跨域的 HTTP 请求，默认情况下是被禁止的，IP（域名）不同、或者端口不同、协议不同（比如 HTTP、HTTPS）都会造成跨域问题。

一般前端的解决方案有：

1、 使用 JSONP 来支持跨域的请求，JSONP 实现跨域请求的原理简单的说，就是动态创建<script>标签，然后利用<script>的 SRC 不受同源策略约束来跨域获取数据。缺点是需要后端配合输出特定的返回信息。

2、 利用反应代理的机制来解决跨域的问题，前端请求的时候先将请求发送到同源地址的后端，通过后端请求转发来避免跨域的访问。

后来 HTML5 支持了 CORS 协议。CORS 是一个 W3C 标准，全称是”跨域资源共享”（Cross-origin resource sharing），允许浏览器向跨源服务器，发出 XMLHttpRequest 请求，从而克服了 AJAX 只能同源使用的限制。它通过服务器增加一个特殊的 Header[Access-Control-Allow-Origin]来告诉客户端跨域的限制，如果浏览器支持 CORS、并且判断 Origin 通过的话，就会允许 XMLHttpRequest 发起跨域请求。

前端使用了 CORS 协议，就需要后端设置支持非同源的请求，SpringBoot 设置支持非同源的请求有两种方式。

第一，配置 CorsFilter。

```
@Configuration
public class GlobalCorsConfig {
    @Bean
    public CorsFilter corsFilter() {
        CorsConfiguration config = new CorsConfiguration();
```

```
            config.addAllowedOrigin("*");
            config.setAllowCredentials(true);
            config.addAllowedMethod("*");
            config.addAllowedHeader("*");
            config.addExposedHeader("*");

            UrlBasedCorsConfigurationSource configSource = new
UrlBasedCorsConfigurationSource();
            configSource.registerCorsConfiguration("/**", config);

            return new CorsFilter(configSource);
        }
}
```

需要配置上述的一段代码。第二种方式稍微简单一些。

第二，在启动类上添加：

```
public class Application extends WebMvcConfigurerAdapter {

    @Override
    public void addCorsMappings(CorsRegistry registry) {

        registry.addMapping("/**")
                .allowCredentials(true)
                .allowedHeaders("*")
                .allowedOrigins("*")
                .allowedMethods("*");

    }
```

```
}
```

JPA 和 Hibernate 有哪些区别？JPA 可以支持动态 SQL 吗？

JPA 本身是一种规范，它的本质是一种 ORM 规范（不是 ORM 框架，因为 JPA 并未提供 ORM 实现，只是制定了规范）因为 JPA 是一种规范，所以，只是提供了一些相关的接口，但是接口并不能直接使用，JPA 底层需要某种 JPA 实现，Hibernate 是 JPA 的一个实现集。

JPA 是根据实体类的注解来创建对应的表和字段，如果需要动态创建表或者字段，需要动态构建对应的实体类，再重新调用 Jpa 刷新整个 Entity。动态 SQL，mybatis 支持的最好，jpa 也可以支持，但是没有 Mybatis 那么灵活。

Spring 、SpringBoot 和 Spring Cloud 的关系？

1、 Spring 最初最核心的两大核心功能 Spring Ioc 和 Spring Aop 成就了 Spring，Spring 在这两大核心的功能上不断的发展，才有了 Spring 事务、Spring Mvc 等一系列伟大的产品，最终成就了 Spring 帝国，到了后期 Spring 几乎可以解决企业开发中的所有问题。

2、 SpringBoot 是在强大的 Spring 帝国生态基础上面发展而来，发明 SpringBoot 不是为了取代 Spring ，是为了让人们更容易的使用 Spring 。

3、 Spring Cloud 是一系列框架的有序集合。它利用 SpringBoot 的开发便利性巧妙地简化了分布式系统基础设施的开发，如服务发现注册、配置中心、消息总线、负载均衡、断路器、数据监控等，都可以用 SpringBoot 的开发风格做到一键启动和部署。

4、 Spring Cloud 是为了解决微服务架构中服务治理而提供的一系列功能的开发框架，并且 Spring Cloud 是完全基于 SpringBoot 而开发，Spring Cloud 利用 SpringBoot 特性整合了开源行业中优秀的组件，整体对外提供了一套在微服务架构中服务治理的解决方案。

5、 用一组不太合理的包含关系来表达它们之间的关系。

6、 Spring ioc/aop > Spring > SpringBoot > Spring Cloud

@SpringBootApplication 注释在内部有什么用处?

作为 Spring 引导文档，@SpringBootApplication 注释等同于同时使用 @Configuration、@EnableAutoConfiguration 和@ComponentScan 及其默认属性。SpringBoot 允许开发人员使用单个注释而不是多个注释。但是，众所周知，Spring 提供了松散耦合的特性，我们可以根据项目需要为每个注释使用这些特性。

如何在不使用 BasePACKAGE 过滤器的情况下排除程序包?

过滤程序包的方法不尽相同。但是弹簧启动提供了一个更复杂的选项，可以在不接触组件扫描的情况下实现这一点。在使用注释@ SpringBootApplication 时，可以使用排除属性。请参阅下面的代码片段：

```
@SpringBootApplication(exclude= {Employee.class})
public class FooAppConfiguration {}
```

如何禁用特定的自动配置类?

若发现任何不愿使用的特定自动配置类，可以使用@EnableAutoConfiguration 的排除属性。

```
//By using "exclude"
@EnableAutoConfiguration(exclude={DataSourceAutoConfiguration.class})
```

另一方面，如果类别不在类路径上，则可以使用 excludeName 类注解，并且指定完全限定名。

```
//By using "excludeName"
@EnableAutoConfiguration(excludeName={Foo.class})
```

此外，SpringBoot 还具有控制排除自动配置类列表的功能，可以通过使用 spring.autoconfigure.exclude property 来实现。可以将其添加到 propertie 应用程序中，并且可以添加逗号分隔的多个类。

```
//By using property file
spring.autoconfigure.exclude=org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConfiguration
```

什么是 Spring Actuator？它有什么优势？

这是 SpringBoot 中最常见的面试问题之一。根据 Spring 文件：

执行器是一个制造术语，指的是移动或控制某物的机械装置。执行机构可以从一个小的变化中产生大量的运动。

众所周知，SpringBoot 提供了许多自动配置特性，帮助开发人员快速开发生产组件。但是，当考虑调试和如何调试，如果出现问题，总是需要分析日志并挖掘应用程序的数据流，检查问题出在何处。因此，Spring Actuator 提供了方便的访问

这些类型的途径。它提供了许多特性，例如创建了什么样的 bean、控制器中的映射、CPU 使用情况等等。它还可以将自动收集和审计健康状况和指标应用到应用程序中。

它提供了一种非常简单的方法来访问少数生产就绪的 REST 端点，并从 Web 获取各种信息。但是通过使用这些端点，你可以做很多事情来查看端点文档。没有必要担心安全问题;如果存在 Spring Security，则默认使用 Spring Security 的内容协商策略保护这些端点。或者，可以在 RequestMatcher 的帮助下配置自定义安全性。

如何启用/禁用执行器?

启用/禁用致动器很容易；最简单的方法是使特性能够将依赖项(Maven/Gradle)添加到 spring-boot-starter-actuator，即启动器。如果不想启用致动器，那么就不要添加依赖项。

Maven 依赖项：

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-actuator</artifactId>
    </dependency>
</dependencies>
```

什么是 Spring Initializer?

这个问题并不难，但面试官总是以此测试候选人的专业知识。

Spring Initializer 是一个网络应用程序，它可以生成一个 SpringBoot 项目，包含快速启动所需的一切。和往常一样，我们需要一个好的项目框架；它有助于你正确创建项目结构/框架。

什么是执行器停机?

关机是允许应用程序正常关机的端点。默认情况下，此功能不启用。你可以在应用程序属性文件中使用 management . endpoint . shut down . enabled = true 来启用此选项。但是该方法请谨慎使用。

是否可以在 Spring boot 中更改嵌入式 Tomcat 服务器的端口?

是的，更改端口是可行的。可以使用 application.properties 文件更改端口。但需要提到“server.port”（即 server.port=8081）。确保项目类路径中有 application.properties；后续工作将由 REST Spring 框架接手。如果提到 server.port=0，那么它将自动分配任何可用的端口。

是否可以在 SpringBoot 中覆盖或替换嵌入式 Tomcat?

是的，可以使用 starter 依赖项将嵌入式 Tomcat 替换为任何其他服务器。可以根据需要使用 SpringBootStarter Jetty 或 SpringBootStarter 作为每个项目的依赖项。

可以在 SpringBoot application 中禁用默认的 Web 服务器吗?

Spring 的主要优势是提供灵活性来构建松散耦合的应用程序。Spring 提供了在快速配置中禁用网络服务器的功能。可以使用应用程序属性来配置网络应用程序类型，例如 spring.main.web-application-type=none.