

# Spring&SpringMVC&#SpringCloud面试题库

---

## #Spring

### 1.什么是Spring框架?

- Spring框架是一个为Java应用程序的开发提供了综合、广泛的基础性支持的Java平台。
- Spring帮助开发者解决了开发中基础性的问题，使得开发人员可以专注于应用程序的开发。
- Spring框架本身亦是按照设计模式精心打造，这使得我们可以在开发环境中安心的集成Spring框架，不必担心Spring是如何在后台进行工作的。

### 2.使用Spring框架能带来哪些好处?

下面列举了一些使用Spring框架带来的主要好处：

- Dependency Injection(DI) 方法使得构造器和 JavaBean properties文件中的依赖关系一目了然。
- 与EJB容器相比较， IoC容器更加趋向于轻量级。这样一来IoC容器在有限的内存和CPU资源的情况下进行应用程序的开发和发布就变得十分有利。
- Spring并没有闭门造车， Spring利用了已有的技术比如ORM框架、 logging框架、 J2EE、 Quartz 和JDK Timer，以及其他视图技术

- Spring框架是按照模块的形式来组织的。由包和类的编号就可以看出其所属的模块，开发者仅仅需要选用他们需要的模块即可
- 要测试一项用Spring开发的应用程序十分简单，因为测试相关的环境代码都已经囊括在框架中了。更加简单的是，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的Web框架亦是一个精心设计的Web MVC框架，为开发者们在web框架的选择上提供了一个除了主流框架比如Struts、过度设计的、不流行web框架的以外的有力选项
- Spring提供了一个便捷的事务管理接口，适用于小型的本地事物处理（比如在单DB的环境下）和复杂的共同事物处理（比如利用JTA的复杂DB环境）
- 

### 3.什么是控制反转(IOC)? 什么是依赖注入(DI)?

- Inversion Of Control ——控制反转，是一种设计思想，程序中需要某个对象时，由原来的新 的过程，变成了由容器来创建、管理和维护组件之间的关系。这样做的好处是 可以大大降低组件之间的耦合度。
- Dependence Injection 依赖注入(依赖注射)。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

## 4. IOC的优点是什么？

- IOC 或 依赖注入把应用的代码量降到最低。它使应用容易测试，单元测试不再需要单例和JNDI查找机制。最小的代价和最小的侵入性使松散耦合得以实现。IOC容器支持加载服务时的饿汉式初始化和懒加载。

## 5.BeanFactory和ApplicationContext有什么区别？

- BeanFactory 可以理解为含有bean集合的工厂类。BeanFactory 包含了种bean的定义，以便在接收到客户端请求时将对应的bean实例化。
- BeanFactory还能在实例化对象的时生成协作类之间的关系。此举将bean自身与bean客户端的配置中解放出来。BeanFactory还包含了bean生命周期的控制，调用客户端的初始化方法 (initialization methods) 和销毁方法 (destruction methods) 。
- 从表面上看，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。

1. 提供了支持国际化的文本消息
2. 统一的资源文件读取方式
3. 已在监听器中注册的bean的事件 以下是三种较常见的 ApplicationContext 实现方式：

- a. ClassPathXmlApplicationContext：从 classpath的XML配置文件中读取上下文，并生成上下文定义。应用程序上下文从程序环境变量中取得。

```
ApplicationContext context = new ClassPathXmlApplicationContext("bean.xml");
```
- b. FileSystemXmlApplicationContext：由文件系统中的XML配置文件读取上下文。

```
ApplicationContext context = new FileSystemXmlApplicationContext("bean.xml");
```
- c. XmlWebApplicationContext：由Web应用的 XML文件读取上下文。

## 6.Spring有几种配置方式？

将Spring配置到应用开发中有以下三种方式：

1. 基于XML的配置
2. 基于注解的配置
3. 基于Java的配置

## 7.如何用基于XML配置的方式配置Spring？

在Spring框架中，依赖和服务需要在专门的配置文件来实现，我常用的XML格式的配置文件。这些配置文件的格式通常用开头，然后一系列的bean定义和专门的应用配置选项组成。

SpringXML配置的主要目的时候是使所有的Spring组件都可以用xml文件的形式来进行配置。这意味着

不会出现其他的Spring配置类型（比如声明的方式或基于Java Class的配置方式）

Spring的XML配置方式是使用被Spring命名空间所支持的一系列的XML标签来实现的。Spring有以下主要的命名空间：context、beans、jdbc、tx、aop、MVC和aso。

```
1 <!-- JSON Support -->
2 <bean name="viewResolver"
3 class="org.springframework.web.servlet.vie
4 w.BeanNameViewResolver"/>
5 <bean name="jsonTemplate"
6 class="org.springframework.web.servlet.vie
7 w.json.MappingJackson2JsonView"/>

<bean id="restTemplate"
8 class="org.springframework.web.client.Rest
9 Template"/>
```

下面这个web.xml仅仅配置了DispatcherServlet，这件最简单的配置便能满足应用程序配置运行时组件的需求。

```
Archetype Created Web Application spring
org.springframework.web.servlet.DispatcherServ
let 1
```

```
1 <servlet-mapping>
2   <servlet-
3     name>spring</servlet-
4     name>
```

```
<url-pattern>/</url-
pattern>
</servlet-mapping>
```

## 8.Spring Bean的作用域之间有什么区别?

Spring容器中的bean可以分为5个范围。所有范围的名称都是自说明的，但是为了避免混淆，还是让我们来解释一下：

1. singleton：这种bean范围是默认的，这种范围确保不管接受到多少个请求，每个容器中只有一个bean的实例，单例的模式由bean factory自身来维护。
2. prototype：原形范围与单例范围相反，为每一个bean请求提供一个实例。
3. request：在请求bean范围内会每一个来自客户端的网络请求创建一个实例，在请求完成以后，bean会失效并被垃圾回收器回收。
4. Session：与请求范围类似，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：global-session和Portlet应用相关。当你的应用部署在Portlet容器中工作时，它包含很多portlet。如果你想要声明让所有的portlet共用全局的存储变量的话，那么这全局变量需要存储在global-session中。全局作用域与Servlet中的session作用域效果相同。

## 9.什么是AOP?

Aspect Oriented Programming 面向切面编程 它可以在不修改原代码的情况下 增加新的功能 或者扩展原有的功能， AOP 关注的重点在切面上， 可以通过配置手段将切面切入到对应的位置。这样组件和组件之间的耦合度会降低 并且可以提供组件的复用度， 提高程序的灵活性。

### AOP 中涉及到的概念

- **Aspect (切面)** : 切面是用来封装共通业务逻辑的 JoinPoint (连接点) : 代表切入的位置 一般代表一个方法信息 Pointcut (切点) : 用来管理连接点的 代表一堆连接点 , 可以看成连接点的集合 (会使用表达式表达 简称切点表达式)
- **Target (目标)** : 将被切入共通业务逻辑的组件叫 目标对象 Proxy (代理 ) : 被增强之后的目标组件 叫代理 (代理的生成技术有JDK代理 和 CGLIB ) Advice9 (通知) : 代表一种时机 (方法之前、 方法执行后、 方法执行前后出现异常, 最终要执行 核心：切面 —— 通知 —— 切点

## 10.Spring主要使用了什么模式?

工厂模式：每个Bean的创建通过方法

单例模式：默认的每个Bean的作用域都是单例

代理模式：关于Aop的实现通过代理模式

## 11.Spring 支持的事务管理类型有哪些?

Spring 支持编程式事务管理和声明式事务管理。许多 Spring 框架的用户选择声明式事务管理，因为这种方式和应用程序的关联较少，因此更加符合轻量级容器的概念。声明式事务管理要优于编程式事务管理，尽管在灵活性方面它弱于编程式事务管理，因为编程式事务允许你通过代码控制业务。

声明式事务又分为两种：

- a、基于XML的声明式事务
- b、基于注解的声明式事务。

## 12. 解释WEB 模块。

Spring的WEB模块是构建在application context 模块基础之上，提供一个适合web应用的上下文。这个模块也包括支持多种面向web的任务，如透明地处理多个文件上传请求和程序级请求参数的绑定到你的业务对象。它也有对Jakarta Struts的支持。

## 13. ApplicationContext通常的实现是什么？

- FileSystemXmlApplicationContext：此容器从一个XML文件中加载beans的定义，XML Bean 配置文件的全路径名必须提供给它的构造函数。
- ClassPathXmlApplicationContext：此容器也从一个XML文件中加载beans的定义，这里，你需要正确设置classpath因为这个容器将在classpath 里找bean配置。

- WebXmlApplicationContext：此容器加载一个 XML文件，此文件定义了一个WEB应用的所有 bean。

## #SpringMVC

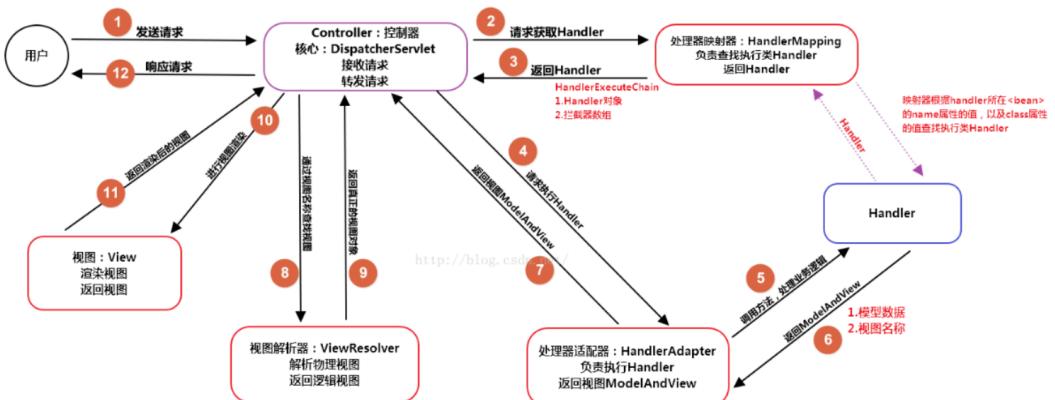
### 1.什么是SpringMVC？简单介绍下你对SpringMVC的理解？

SpringMVC是一个基于Java的实现了MVC设计模式的请求驱动类型的轻量级Web框架，通过把 Model, View, Controller分离，将web层进行职责解耦，把复杂的web应用分成逻辑清晰的几部分，简化开发，减少出错，方便组内开发人员之间的配合。

### 2.SpringMVC的工作流程

- 用户发送请求至前端控制器DispatcherServlet
- DispatcherServlet收到请求调用 HandlerMapping处理器映射器
- 处理器映射器根据请求url找到具体的处理器，生成处理器对象及处理器拦截器(如果有则生成)一并返回给DispatcherServlet。
- DispatcherServlet通过HandlerAdapter处理器适配器调用处理器
- 执行处理器(Controller，也叫后端控制器)
- Controller执行完成返回ModelAndView
- HandlerAdapter将controller执行结果 ModelAndView返回给DispatcherServlet

- DispatcherServlet将ModelAndView传给ViewReslover视图解析器
- ViewReslover解析后返回具体View
- DispatcherServlet对View进行渲染视图（即将模型数据填充至视图中）
- DispatcherServlet响应用户



### 3. SpringMVC的优点:

1. 可以支持各种视图技术,而不仅仅局限于JSP;
2. 与Spring框架集成 (如IoC容器、AOP等) ;
3. 清晰的角色分配：前端控制器  
(dispatcherServlet), 请求到处理器映射  
(handlerMapping), 处理器适配器  
(HandlerAdapter), 视图解析器  
(ViewResolver) 。
4. 支持各种请求资源的映射策略。

### 4. 注解原理

注解本质是一个继承了Annotation的特殊接口，其具体实现类是Java运行时生成的动态代理类。我们通过反射获取注解时，返回的是Java运行时生成的

动态代理对象。通过代理对象调用自定义注解的方法，会最终调用AnnotationInvocationHandler的invoke方法。该方法会从memberValues这个Map中索引出对应的值。而memberValues的来源是Java常量池。

## 5.SpringMVC和struts2的区别有哪些？

1. SpringMVC的入口是一个servlet即前端控制器（DispatchServlet），而struts2入口是一个filter过滤器（StrutsPrepareAndExecuteFilter）。
2. SpringMVC是基于方法开发(一个url对应一个方法)，请求参数传递到方法的形参，可以设计为单例或多例(建议单例)，struts2是基于类开发，传递参数是通过类的属性，只能设计为多例。
3. Struts采用值栈存储请求和响应的数据，通过OGNL存取数据，SpringMVC通过参数解析器是将request请求内容解析，并给方法形参赋值，将数据和视图封装成 ModelAndView 对象，最后又将 ModelAndView 中的模型数据通过 request 域传输到页面。Jsp视图解析器默认使用jstl。

## 6.SpringMVC怎么样设定重定向和转发的？

1. 转发：在返回值前面加”forward:”，譬如”forward:user.do?name=method4”
2. 重定向：在返回值前面加”redirect:”，譬如”redirect:http://www.baidu.com”

## 7.SpringMVC怎么和AJAX相互调用的？

通过Jackson框架就可以把Java里面的对象直接转化成Js可以识别的Json对象。

具体步骤如下：

1. 加入Jackson.jar
2. 在配置文件中配置json的映射
3. 在接受Ajax方法里面可以直接返回Object,List等，但方法前面要加上@ResponseBody注解
- 4.

## 8.如何解决POST请求中文乱码问题，GET的又如何处理呢？

### (1) 解决post请求乱码问题：

在web.xml中配置一个CharacterEncodingFilter过滤器，设置成utf-8；

```
<filter>
    <filter-
name>CharacterEncodingFilter</filter-name>
    <filter-
class>org.springframework.web.filter.CharacterE
ncodingFilter</filter-class>
    <init-param>
        <param-name>encoding</param-
name>
        <param-value>utf-8</param-value>
    </init-param>
```

```
</filter>  
<filter-mapping>  
    <filter-  
name>CharacterEncodingFilter</filter-name>  
    <url-pattern>/*</url-pattern>  
</filter-mapping>
```

## (2) get请求中文参数出现乱码解决方法有两个：

1. 修改tomcat配置文件添加编码与工程编码一致，  
如下：

```
<ConnectorURIEncoding="utf-8"  
connectionTimeout="20000" port="8080"  
protocol="HTTP/1.1" redirectPort="8443"/>
```

1. 另外一种方法对参数进行重新编码：

```
String userName = new  
String(request.getParamter("userName").getByt  
es("ISO8859-1"),"utf-8")
```

ISO8859-1是tomcat默认编码，需要将tomcat编码  
后的内容按utf-8编码。

## 9.SpringMVC的异常处理？

可以将异常抛给Spring框架，由Spring框架来处  
理；我们只需要配置简单的异常处理器，在异常处  
理器中添视图页面即可。

## 10.SpringMVC的控制器是不是单例模式,如果是,有什么问题,怎么解决?

是单例模式,所以在多线程访问的时候有线程安全问题,不要用同步,会影响性能的,解决方案是在控制器里面不能写字段。

## 11.SpringMVC常用的注解有哪些?

@RequestMapping: 用于处理请求 url 映射的注解, 可用于类或方法上。用于类上, 则表示类中的所有响应请求的方法都是以该地址作为父路径。

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

@ResponseBody: 注解实现将controller方法返回对象转化为json对象响应给客户。

## 12.SpringMVC中的控制器的注解一般用那个,有没有别的注解可以替代?

一般用@Controller注解,也可以使用@RestController,@RestController注解相当于@ResponseBody + @Controller,表示是表现层,除此之外,一般不用别的注解代替。

## 13.如果在拦截请求中,我想拦截get方式提交的方法,怎么配置?

可以在@RequestMapping注解里面加上method=RequestMethod.GET。

## **14.怎样在方法里面得到Request,或者Session?**

直接在方法的形参中声明request, SpringMVC就自动把request对象传入。

## **15.如果想在拦截的方法里面得到从前台传入的参数,怎么得到?**

直接在形参里面声明这个参数就可以,但必须名字和传过来的参数一样。

## **16.如果前台有很多个参数传入,并且这些参数都是一个对象的,那么怎么样快速得到这个对象?**

直接在方法中声明这个对象, SpringMVC就自动会把属性赋值到这个对象里面。

## **17.SpringMVC中函数的返回值是什么?**

返回值可以有很多类型,有String, ModelAndView。 ModelAndView类把视图和数据都合并的一起的,但一般用String比较好。

## **18.SpringMVC用什么对象从后台向前台传递数据的?**

通过ModelMap对象,可以在这个对象里面调用put方法,把对象加到里面,前台就可以通过el表达式拿到。

## **19.怎么样把ModelMap里面的数据放入Session里面?**

可以在类上面加上@SessionAttributes注解,里面包含的字符串就是要放入session里面的key。

## 20.SpringMVC里面拦截器是怎么写的

有两种写法,一种是实现HandlerInterceptor接口,  
另外一种是继承适配器类, 接着在接口方法当中,  
实现处理逻辑; 然后在SpringMVC的配置文件中配  
置拦截器即可:

```
<!-- 配置SpringMVC的拦截器 -->
<MVC:interceptors>
    <!-- 配置一个拦截器的Bean就可以了 默认是
对所有请求都拦截 -->
    <bean id="myInterceptor"
        class="com.zwp.action.MyHandlerInterceptor">
    </bean>
    <!-- 只针对部分请求拦截 -->
    <MVC:interceptor>
        <MVC:mapping path="/modelMap.do">
        />
        <bean
            class="com.zwp.action.MyHandlerInterceptorAdapter" />
        </MVC:interceptor>
    </MVC:interceptors>
```

## #Spring Cloud

### 1.什么是Spring Cloud?

Spring Cloud流应用程序启动器是基于Spring Boot的Spring集成应用程序，提供与外部系统的集成。

Spring Cloud Task，一个生命周期短暂的微服务框架，用于快速构建执行有限数据处理的应用程序

## 2.Spring Cloud 的核心组件有哪些？

Eureka：服务注册与发现。

Feign：基于动态代理机制，根据注解和选择的机器，拼接请求 url 地址，发起请求。

Ribbon：实现负载均衡，从一个服务的多台机器中选择一台。

Hystrix：熔断 提供线程池，不同的服务走不同的线程池，实现了不同服务调用的隔离，避免了服务雪崩的问题。

Zuul：网关管理，由 Zuul 网关转发请求给对应的服务。

## 3.微服务的优点缺点？

优点：

1. 每个服务直接且足够内聚，代码容易理解
2. 开发效率高，一个服务只做一件事，适合小团队开发
3. 松耦合，有功能意义的服务
4. 可以用不同语言开发，面向接口编程
5. 易于第三方集成
6. 微服务只是业务逻辑的代码，不会和HTML,CSS或其他界面结合。

## 7. 可以灵活搭配，连接公共库/连接独立库

### 缺点：

1. 分布式系统的责任性
2. 多服务运维难度加大
3. 系统部署依赖，服务间通信成本，数据一致性，系统集成测试，性能监控

### 4. Spring Cloud 和Dubbo区别？

1. 服务调用方式 Dubbo是RPC Spring Cloud Rest API
2. 注册中心,Dubbo 是zookeeper Spring Cloud是 eureka，也可以是zookeeper
3. 服务网关,Dubbo本身没有实现，只能通过其他第三方技术整合，Spring Cloud有Zuul路由网关，作为路由服务器，进行消费者的请求分发, Spring Cloud支持断路器，与git完美集成配置文件支持版本控制，事物总线实现配置文件的更新与服务自动装配等等一系列的微服务架构要素

### 5. REST 和RPC对比

1. RPC主要的缺陷是服务提供方和调用方式之间的依赖太强，需要对每一个微服务进行接口的定义，并通过持续继承发布，严格版本控制才不会出现冲突。
2. REST是轻量级的接口，服务的提供和调用不存在代码之间的耦合，只需要一个约定进行规范。

## 6.你所知道的微服务技术栈?

维度(Spring Cloud)

服务开发: springboot spring SpringMVC

服务配置与管理:Netflix公司的Archaius ,阿里的Diamond

服务注册与发现:Eureka,Zookeeper

服务调用:Rest RPC gRpc

服务熔断器:Hystrix

服务负载均衡:Ribbon Nginx

服务接口调用:Fegin

消息队列:Kafka Rabbitmq activemq

服务配置中心管理:Spring CloudConfig

服务路由 (API网关) Zuul

事件消息总线:Spring Cloud Bus

## 7.负载均衡的意义是什么?

在计算中，负载均衡可以改善跨计算机，计算机集群，网络链接，中央处理单元或磁盘驱动器等多种计算资源的工作负载分布。负载均衡旨在优化资源使用，最大吞吐量，最小响应时间并避免任何单一资源的过载。使用多个组件进行负载均衡而不是单个组件可能会通过冗余来提高可靠性和可用性。负载平衡通常涉及专用软件或硬件，例如多层交换机或域名系统服务进程。

## 8.微服务之间是如何独立通讯的?

远程调用(比如Feign调用，直接通过远程过程调用  
来访问别的service  
消息中间件

## 9.Spring Cloud如何实现服务的注册？

- 1.服务发布时，指定对应的服务名,将服务注册到 注册中心(eureka zookeeper)
- 2.注册中心加@EnableEurekaServer,服务用 @EnableDiscoveryClient，然后用Ribbon或Feign 进行服务直接的调用发现。

## 10.Eureka和Zookeeper区别

- Eureka取CAP的AP，注重可用性，Zookeeper取 CAP的CP注重一致性
- Zookeeper在选举期间注册服务瘫痪，虽然服务最终会恢复，但选举期间不可用
- eureka的自我保护机制，会导致一个结果就是不会再从注册列表移除因长时间没收到心跳而过期的服务。依然能接受新服务的注册和查询请求，但不会被同步到其他节点。不会服务瘫痪
- Zookeeper有Leader和Follower角色，Eureka各个节点平等。
- Zookeeper采用过半数存活原则，Eureka采用自我保护机制解决分区问题
- eureka本质是一个工程，Zookeeper只是一个进程

## 11.Eureka自我保护机制是什么？

当Eureka Server 节点在短时间内丢失了过多实例的连接时（比如网络故障或频繁启动关闭客户端）节点会进入自我保护模式，保护注册信息，不再删除注册数据，故障恢复时，自动退出自我保护模式。

## 12.什么是服务熔断？什么是服务降级？

服务直接的调用，比如在高并发情况下出现进程阻塞，导致当前线程不可用，慢慢的全部线程阻塞，导致服务器雪崩。

服务熔断：相当于保险丝，出现某个异常，直接熔断整个服务，而不是一直等到服务超时。通过维护一个自己的线程池，当线程到达阈值的时候就启动服务降级，如果其他请求继续访问就直接返回 fallback的默认值。

## 13.什么是Ribbon？

Ribbon是一个负载均衡客户端，可以很好的控制http和tcp的一些行为。Feign默认集成了Ribbon。

## 14.什么是Feign？它的优点是什么？

1. Feign采用的是基于接口的注解
2. Feign整合了Ribbon，具有负载均衡的能力
3. 整合了Hystrix，具有熔断的能力

使用：

1. 添加pom依赖。
2. 启动类添加@EnableFeignClients
3. 定义一个接口@FeignClient(name="xxx")指定调用哪个服务

## 15.Ribbon和Feign的区别?

1. Ribbon都是调用其他服务的，但方式不同
2. 启动类注解不同， Ribbon是@RibbonClient  
Feign的是@EnableFeignClients
3. 服务指定的位置不同， Ribbon是在  
@RibbonClient注解上声明， Feign则是在定义抽象方法的接口中使用@FeignClient声明
4. 调用方式不同， Ribbon需要自己构建http请求，  
模拟http请求然后使用RestTemplate发送给其他服务， 步骤相当繁琐。 Feign需要将调用的方法定义成抽象方法即可

## 16.什么是Spring Cloud Bus?

- spring cloud bus 将分布式的节点用轻量的消息代理连接起来，它可以用于广播配置文件的更改或者服务直接的通讯，也可用于监控。
- 如果修改了配置文件，发送一次请求，所有的客户端便会重新读取配置文件。

使用:

1. 添加依赖
2. 配置rabbimq

## 17.Spring Cloud断路器作用?

1. 当一个服务调用另一个服务由于网络原因或自身原因出现问题，调用者就会等待被调用者的响应当更多的服务请求到这些资源导致更多的请求等待，发生连锁效应（雪崩效应）
2. 断路器有完全打开状态:一段时间内 达到一定的次数无法调用 并且多次监测没有恢复的迹象 断路器完全打开 那么下次请求就不会请求到该服务
  - 半开:短时间内 有恢复迹象 断路器会将部分请求发给该服务，正常调用时 断路器关闭
  - 关闭：当服务一直处于正常状态 能正常调用

## 18.什么是Spring CloudConfig?

在分布式系统中，由于服务数量巨多，为了方便服务配置文件统一管理，实时更新，所以需要分布式配置中心组件。在Spring Cloud中，有分布式配置中心组件spring cloud config，它支持配置服务放在配置服务的内存中（即本地），也支持放在远程Git仓库中。在spring cloud config 组件中，分两个角色，一是config server，二是config client。

使用：

- 添加pom依赖
- 配置文件添加相关配置
- 启动类添加注解@EnableConfigServer

## 19.Spring Cloud Gateway?

Spring Cloud Gateway是Spring Cloud官方推出的第二代网关框架，取代Zuul网关。网关作为流量的，在微服务系统中有着非常作用，网关常见的功能有路由转发、权限校验、限流控制等作用。

使用了一个RouteLocatorBuilder的bean去创建路由，除了创建路由RouteLocatorBuilder可以让你添加各种predicates和filters，predicates断言的意思，顾名思义就是根据具体的请求的规则，由具体的route去处理，filters是各种过滤器，用来对请求做各种判断和修改。

## 20.架构

在微服务架构中，需要几个基础的服务治理组件，包括服务注册与发现、服务消费、负载均衡、断路器、智能路由、配置管理等，由这几个基础组件相互协作，共同组建了一个简单的微服务系统

在Spring Cloud微服务系统中，一种常见的负载均衡方式是，客户端的请求首先经过负载均衡（Zuul、Ngnix），再到达服务网关（Zuul集群），然后再到达具体的服。服务统一注册到高可用的服务注册中心集群，服务的所有的配置文件由配置服务管理，配置服务的配置文件放在git仓库，方便开发人员随时改配置。

## 21.什么是Hystrix?

**防雪崩利器：**具备服务降级，服务熔断，依赖隔离，监控（Hystrix Dashboard）

服务降级：

双十一提示：哎哟喂，被挤爆了

app秒杀：网络开小差了，请稍后再试

优先核心服务，非核心服务不可用或弱可用。

通过HystrixCommand注解指定。

fallbackMethod(回退函数)中具体实现降级逻辑。