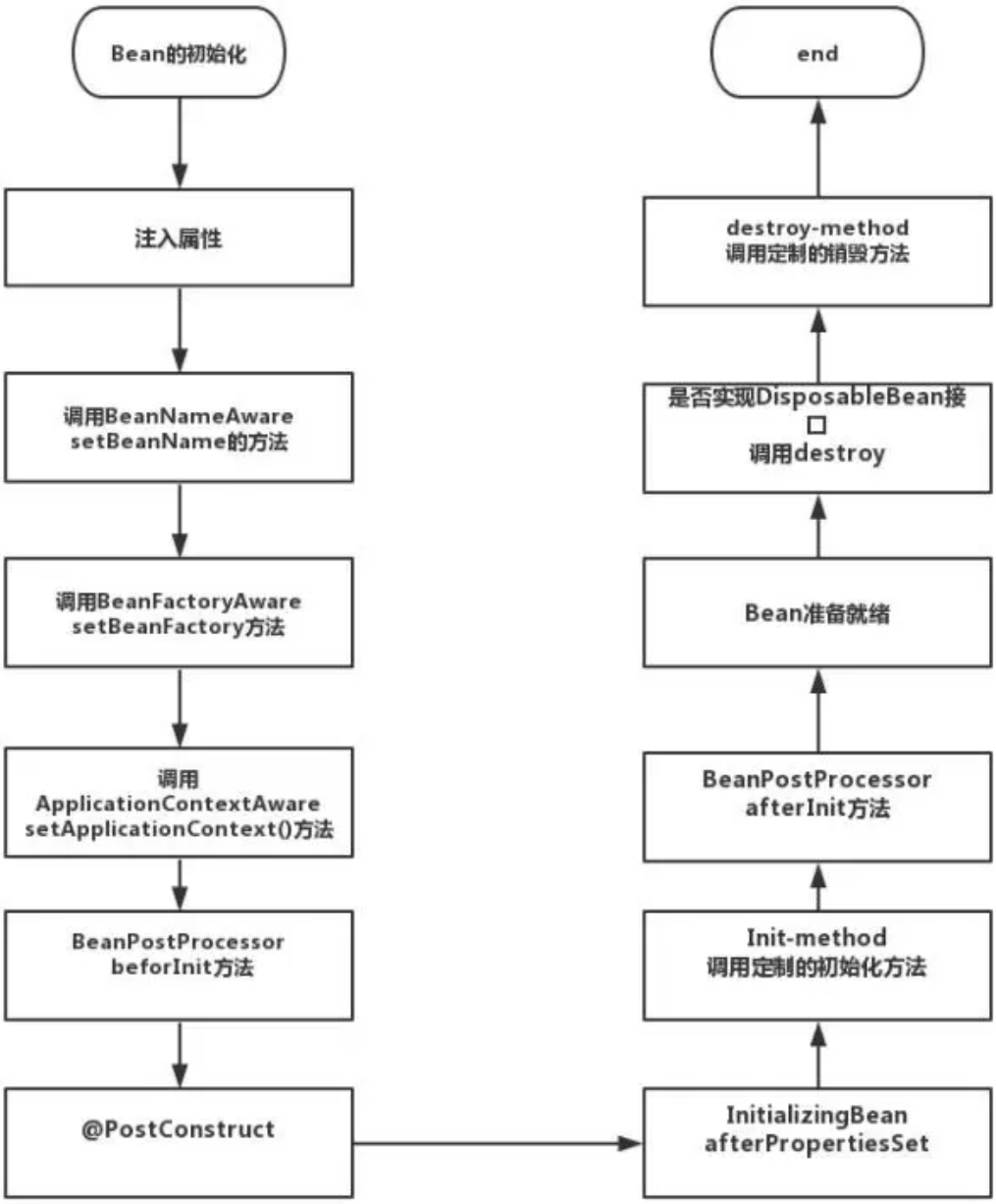
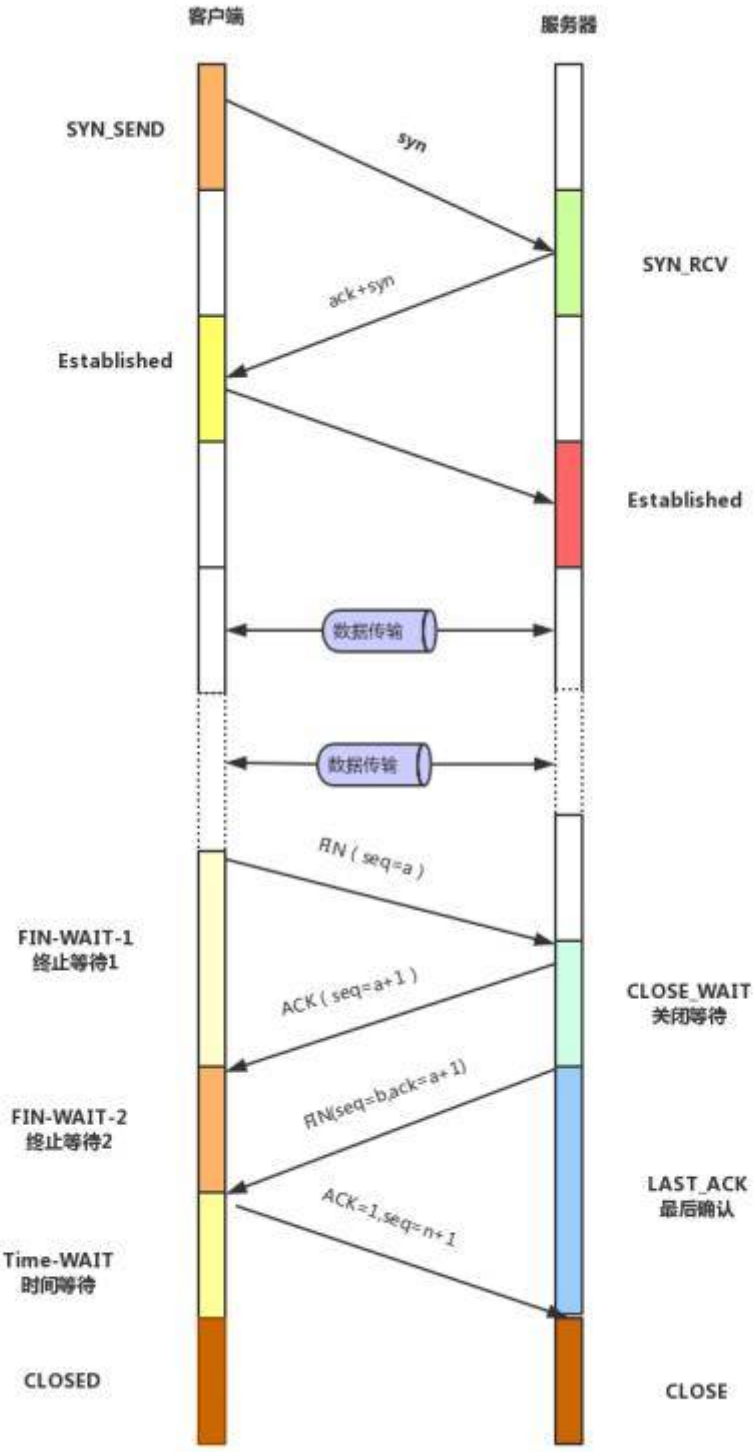
Spring的生命周期



1. 首先容器启动后，对bean进行初始化
2. 按照bean的定义，注入属性
3. 检测该对象是否实现了xxxAware接口，并将相关的xxxAware实例注入给bean，如BeanNameAware等
4. 以上步骤，bean对象已正确构造，通过实现BeanPostProcessor接口，可以再进行一些自定义方法处理。如:postProcessBeforeInitialzation。
5. BeanPostProcessor的前置处理完成后，可以实现postConstruct，afterPropertiesSet,init-method等方法，增加我们自定义的逻辑
6. 通过实现BeanPostProcessor接口，进行postProcessAfterInitialzation后置处理
7. 接着Bean准备好被使用啦。
8. 容器关闭后，如果Bean实现了DisposableBean接口，则会回调该接口的destroy()方法

9. 通过给destroy-method指定函数，就可以在bean销毁前执行指定的逻辑

TCP三次握手，四次挥手



三次握手

- 第一次握手(SYN=1, seq=x)，发送完毕后，客户端进入 SYN_SEND 状态
- 第二次握手(SYN=1, ACK=1, seq=y, ACKnum=x+1)， 发送完毕后，服务器端进入 SYN_RCVD 状态。

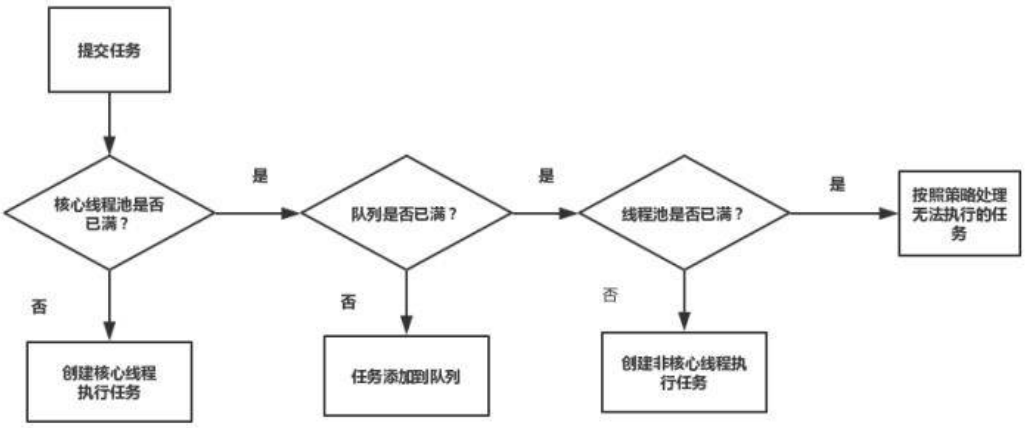
- 第三次握手(ACK=1，ACKnum=y+1)，发送完毕后，客户端进入 ESTABLISHED 状态，当服务器端接收到这个包时，也进入 ESTABLISHED 状态，TCP 握手，即可以开始数据传输。

四次挥手

- 第一次挥手(FIN=1，seq=a)，发送完毕后，客户端进入 FIN_WAIT_1 状态
- 第二次挥手(ACK=1，ACKnum=a+1)，发送完毕后，服务器端进入 CLOSE_WAIT 状态，客户端接收到这个确认包之后，进入 FIN_WAIT_2 状态
- 第三次挥手(FIN=1，seq=b)，发送完毕后，服务器端进入 LAST_ACK 状态，等待来自客户端的最后一个ACK。
- 第四次挥手(ACK=1，ACKnum=b+1)，客户端接收到来自服务器端的关闭请求，发送一个确认包，并进入 TIME_WAIT状态，等待了某个固定时间（两个最大段生命周期，2MSL，2 Maximum Segment Lifetime）之后，没有收到服务器端的 ACK ，认为服务器端已经正常关闭连接，于是自己也关闭连接，进入 CLOSED 状态。服务器端接收到这个确认包之后，关闭连接，进入 CLOSED 状态。

线程池执行流程图

线程池：一种线程使用模式。线程过多会带来调度开销，进而影响缓存局部性和整体性能。而线程池维护着多个线程，等待着监督管理者分配可并发执行的任务，这避免了在处理短时间任务时创建与销毁线程的代价。线程池执行流程是每个开发必备的。



执行流程

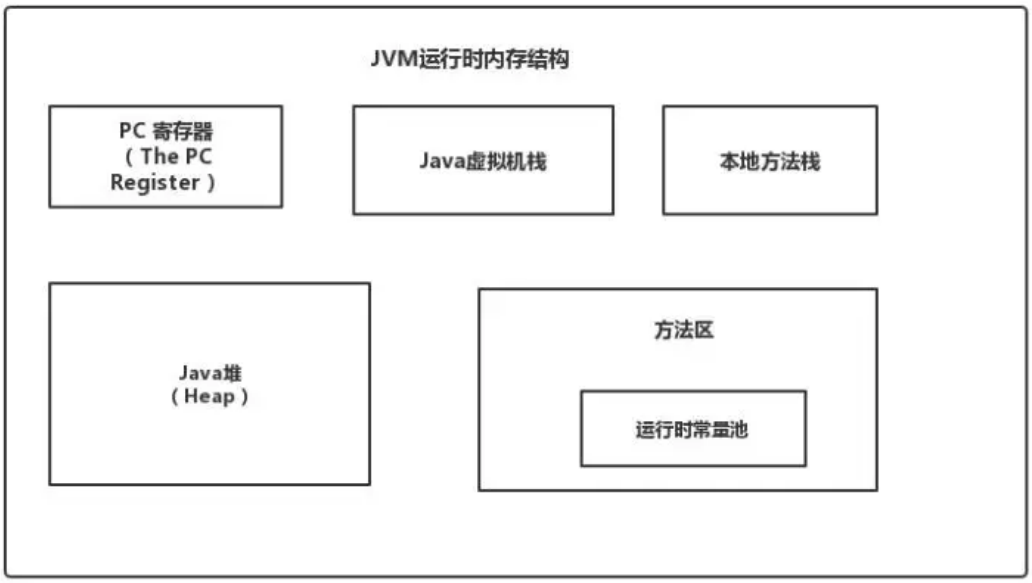
- 提交一个任务，线程池里存活的核心线程数小于线程数corePoolSize时，线程池会创建一个核心线程去处理提交的任务。
- 如果线程池核心线程数已满，即线程数已经等于corePoolSize，一个新提交的任务，会被放进任务队列workQueue排队等待执行。
- 当线程池里面存活的线程数已经等于corePoolSize了,并且任务队列workQueue也满，判断线程数是否达到maximumPoolSize，即最大线程数是否已满，如果没到达，创建一个非核心线程执行提交的任务。
- 如果当前的线程数达到了maximumPoolSize，还有新的任务过来的话，直接采用拒绝策略处理。

JDK提供了四种拒绝策略处理类

- AbortPolicy(抛出一个异常，默认的)
- DiscardPolicy(直接丢弃任务)

- DiscardOldestPolicy（丢弃队列里最老的任务，将当前这个任务继续提交给线程池）
- CallerRunsPolicy（交给线程池调用所在的线程进行处理）

JVM内存结构



程序计数器（PC 寄存器）

程序计数器是一块较小的内存空间，可以看作当前线程所执行的字节码的行号指示器。在虚拟机的模型里，字节码解释器工作时就是通过改变这个计数器的值来选取下一条需要执行的字节码指令，分支、循环、异常处理、线程恢复等基础功能都需要依赖计数器完成。

Java虚拟机栈

- 与程序计数器一样，Java虚拟机栈也是线程私有的，它的生命周期与线程相同
- 每个方法被执行的时候都会创建一个"栈帧",用于存储局部变量表(包括参数)、操作数栈、动态链接、方法出口等信息。每个方法被调用到执行完的过程，就对应着一个栈帧在虚拟机栈中从入栈到出栈的过程。
- 局部变量表存放各种基本数据类型boolean、byte、char、short等

本地方法栈

与虚拟机栈基本类似，区别在于虚拟机栈为虚拟机执行的java方法服务，而本地方法栈则是为Native方法服务。

Java堆

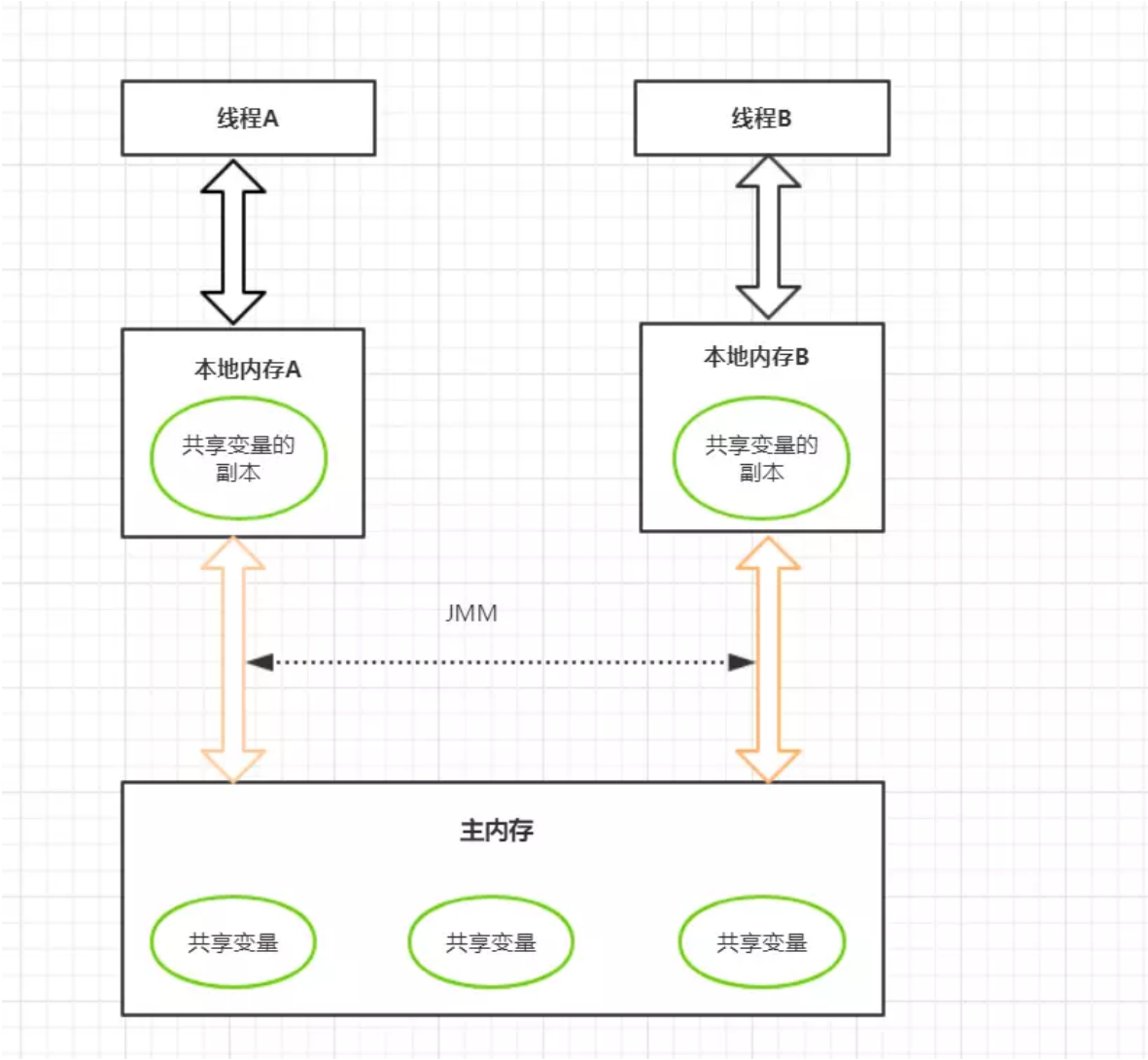
- GC堆是java虚拟机所管理的内存中最大的一块内存区域，也是被各个线程共享的内存区域，在JVM启动时创建。
- 其大小通过-Xms(最小值)和-Xmx(最大值)参数设置，-Xms为JVM启动时申请的最小内存，-Xmx为JVM可申请的最大内存。
- 由于现在收集器都是采用分代收集算法，堆被划分为新生代和老年代。新生代由S0和S1构成，可通过-Xmn参数来指定新生代的大小。

- 所有对象实例以及数组都在堆上分配。
- Class文件中除了有类的版本、字段、方法、接口等描述信息外，还有一项信息是常量池，用于存放编译器生成的各种符号引用，这部分内容将在类加载后放到方法区的运行时常量池中。

方法区

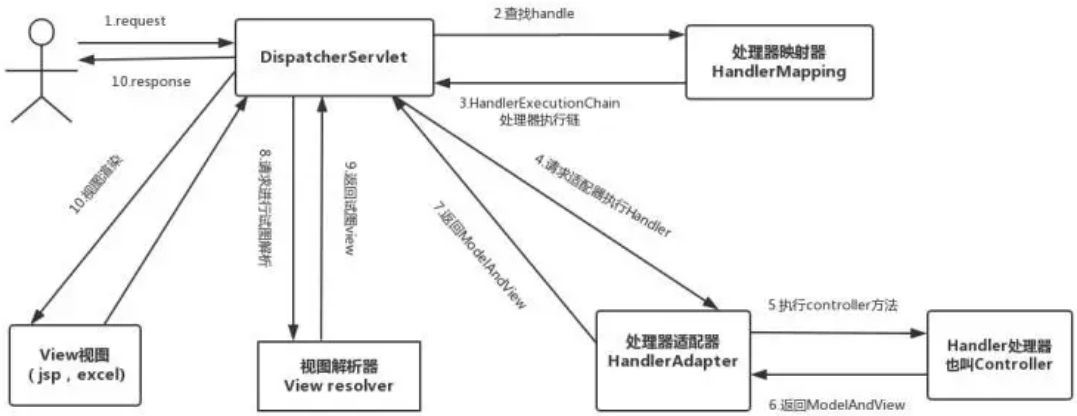
- 也称”永久代“，它用于存储虚拟机加载的类信息、常量、静态变量、是各个线程共享的内存区域。可以通过-XX:PermSize 和 -XX:MaxPermSize 参数限制方法区的大小。
- 运行时常量池：是方法区的一部分，其中的主要内容来自于JVM对Class的加载。
- Class文件中除了有类的版本、字段、方法、接口等描述信息外，还有一项信息是常量池，用于存放编译器生成的各种符号引用，这部分内容将在类加载后放到方法区的运行时常量池中。

Java内存模型



- Java的多线程之间是通过共享内存进行通信的，在通信过程中会存在一系列如可见性、原子性、顺序性等问题，而JMM就是围绕着多线程通信以及与其相关的一系列特性而建立的模型。JMM定义了一些语法集，这些语法集映射到Java语言中就是volatile、synchronized等关键字。
- Java内存模型规定了所有的变量都存储在主内存中，每条线程还有自己的工作内存，线程的工作内存中保存了该线程中是用到的变量的主内存副本拷贝，线程对变量的所有操作都必须在工作内存中进行，而不能直接读写主内存。不同的线程之间也无法直接访问对方工作内存中的变量，线程间变量的传递均需要自己的工作内存和主存之间进行数据同步进行。

springMVC执行流程图



1. User向服务器发送request,前端控制Servelt DispatcherServlet捕获;
2. DispatcherServlet对请求URL进行解析，调用HandlerMapping获得该Handler配置的所有相关的对象，最后以HandlerExecutionChain对象的形式返回.
3. DispatcherServlet 根据获得的Handler，选择一个合适的HandlerAdapter.
4. 提取Request中的模型数据，填充Handler入参，开始执行Handler（Controller)
5. Handler执行完成后，返回一个ModelAndView对象到DispatcherServlet
6. 根据返回的ModelAndView，选择一个适合的ViewResolver
7. ViewResolver 结合Model和View，来渲染视图
8. 将渲染结果返回给客户端。

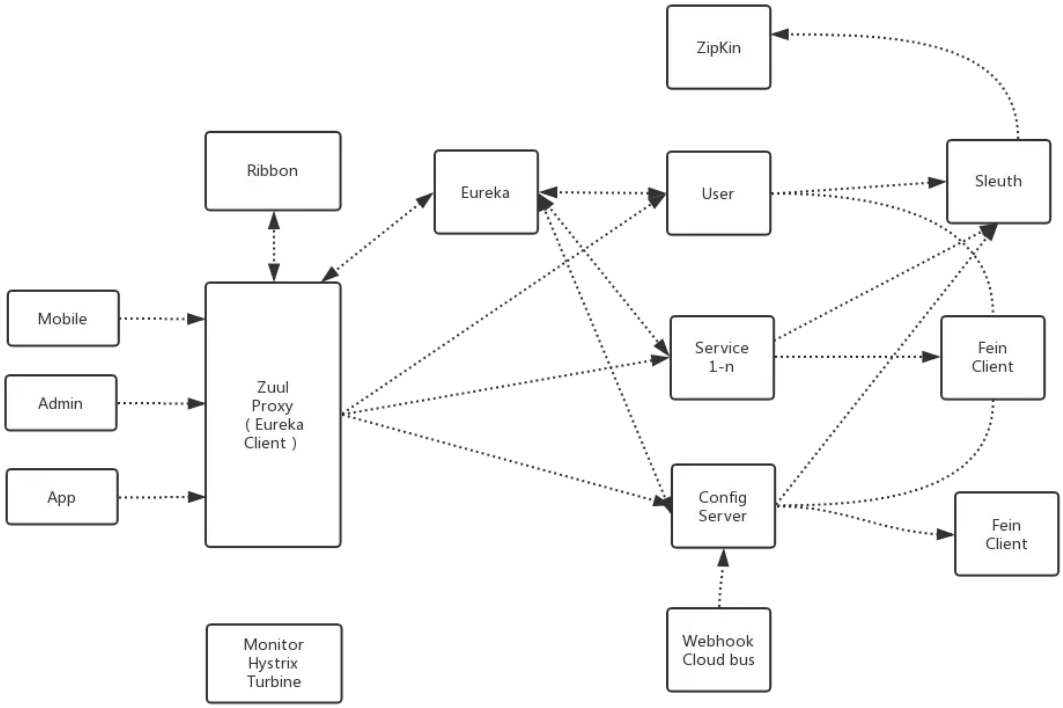
JDBC执行流程

JDBC执行流程：

- 连接数据源
- 为数据库传递查询和更新指令
- 处理数据库响应并返回的结果

spring cloud组件架构

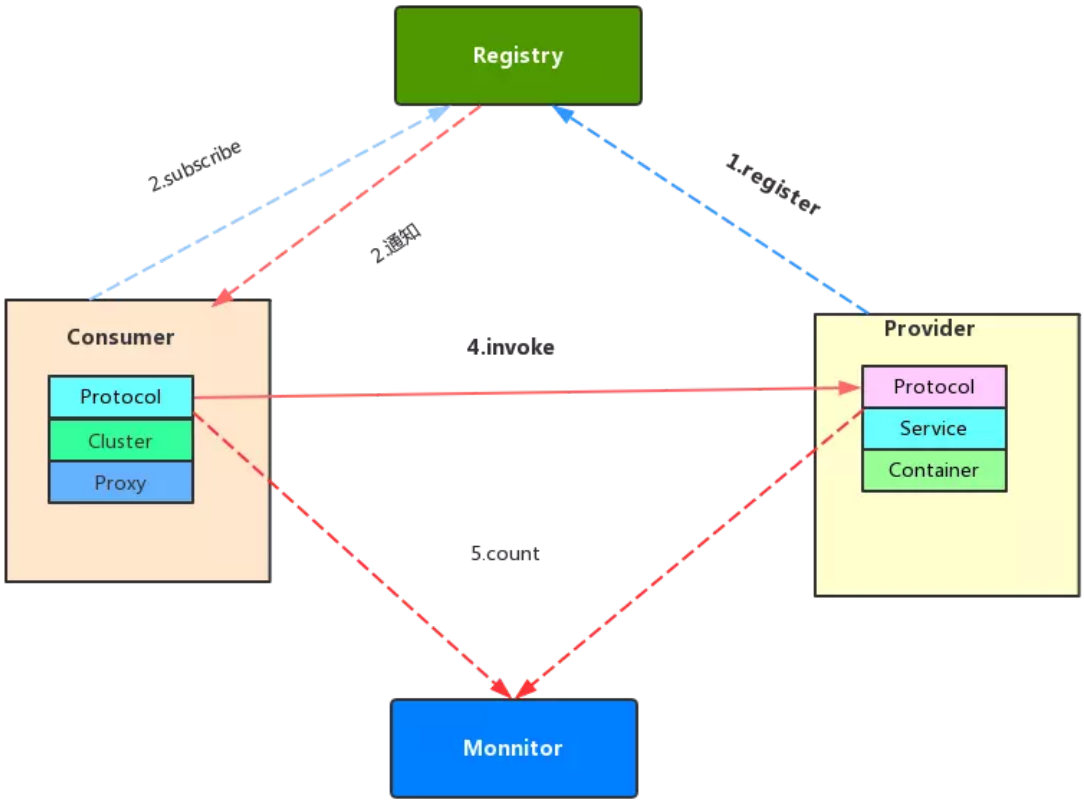
Spring Cloud是一个基于Spring Boot实现的云原生应用开发工具，它为基于JVM的云原生应用开发中涉及的配置管理、服务发现、熔断器、智能路由、微代理、控制总线、分布式会话和集群状态管理等操作提供了一种简单的开发方式。



- Eureka 负责服务的注册与发现。
- Hystrix 负责监控服务之间的调用情况，起到熔断,降级作用。
- Spring Cloud Config 提供了统一的配置中心服务。
- 所有对外的请求和服务，我们都通过Zuul来进行转发，起到 API 网关的作用
- 最后我们使用 Sleuth+Zipkin 将所有的请求数据记录下来，方便我们进行后续分析。
- Spring Cloud Ribbon是基于Netflix Ribbon实现的一套客户端负载均衡的工具。
- 它是一个基于HTTP和TCP的客户端负载均衡器。
- Feign是一个声明式的Web Service客户端，它的目的就是让Web Service调用更加简单。

dubbo 调用

Dubbo是一个分布式服务框架，致力于提供高性能和透明化的远程服务调用方案，这容易和负载均衡弄混，负载均衡是对外提供一个公共地址，请求过来时通过轮询、随机等，路由到不同server。



- Provider: 暴露服务的服务提供方。
- Consumer: 调用远程服务的服务消费方。
- Registry: 服务注册与发现的注册中心。
- Monitor: 统计服务的调用次调和调用时间的监控中心。
- Container: 服务运行容器。