1. 微服务：
   1. feign远程调用服务的步骤：
2. 引入依赖
3. 开启注解（yml配置文件无需配置）
4. 写feignClient接口，接口里面写需要调用的远程方法
5. 在需要进行远程调用的方法里注入该接口，并调用对应的api接口方法
6. hystric熔断(需要yml配置文件中开启)，写一个feignClient接口的实现类（记得加@component），在里面写fallback的业务逻辑；
7. 设计模式
   1. 策略模式：对象的行为，在不同的环境下，有不同的实现；符合“开闭原则”，可以在不修改原有代码的基础上替换、添加新的策略；不同的策略可以相互替换；客户端自己决定在什么情况下使用什么具体策略角色；与状态模式区别：使用策略模式时，客户端手动选择策略，使用状态模式时，其行为是根据状态是自动切换的；
   2. 状态模式：**一**个对象的行为根据其内部状态的改变自动变化，结构与策略模式基本一致；与策略模式区别：使用策略模式时，客户端手动选择策略，使用状态模式时，其行为是根据状态是自动切换的。其内部状态改变时，它的行为（方法）也跟着改变，看起来就像修改了类的方法
   3. 备忘录模式：；
   4. 命令模式（常与备忘录模式结合使用）：将请求封装成命令对象，请求的具体执行由命令接收者执行；命令发送者与命令执行者解耦；每一个命令都是一个操作：请求的一方发出请求，要求执行一个操作；接收的一方收到请求，并执行操作。
   5. 适配器模式：
   6. 组合模式：组合模式为处理树形结构提供了一种较为完美的解决方案，它描述了如何将容器和叶子进行递归组合，使得用户在使用时无须对它们进行区分，可以一致地对待容器和叶子。组合多个对象形成树形结构以表示具有“整体—部分”关系的层次结构。组合模式对单个对象（即叶子对象）和组合对象（即容器对象）的使用具有一致性。
   7. 外观模式：
   8. 单例模式：
   9. 工厂方法模式：
   10. 抽象工厂模式：
   11. 装饰者模式：
   12. 代理模式：
   13. 迭代器模式：
   14. 观察者模式：
   15. 原型模式：
   16. 职责链模式：避免请求发送者与接收者耦合在一起，让多个对象都有可能接收请求，将这些对象连接成一条链，并且沿着这条链传递请求，直到有对象处理它为止。职责链模式是一种对象行为型模式。
   17. 桥接模式：桥接模式是一种很实用的结构型设计模式，如果软件系统中某个类存在两个独立变化的维度，通过该模式可以将这两个维度分离出来，使两者可以独立扩展，让系统更加符合“单一职责原则”。 桥接模式用一种巧妙的方式处理多层继承存在的问题，用抽象关联取代了传统的多层继承，将类之间的静态继承关系转换为动态的对象组合关系，使得系统更加灵活，并易于扩展，同时有效控制了系统中类的个数。