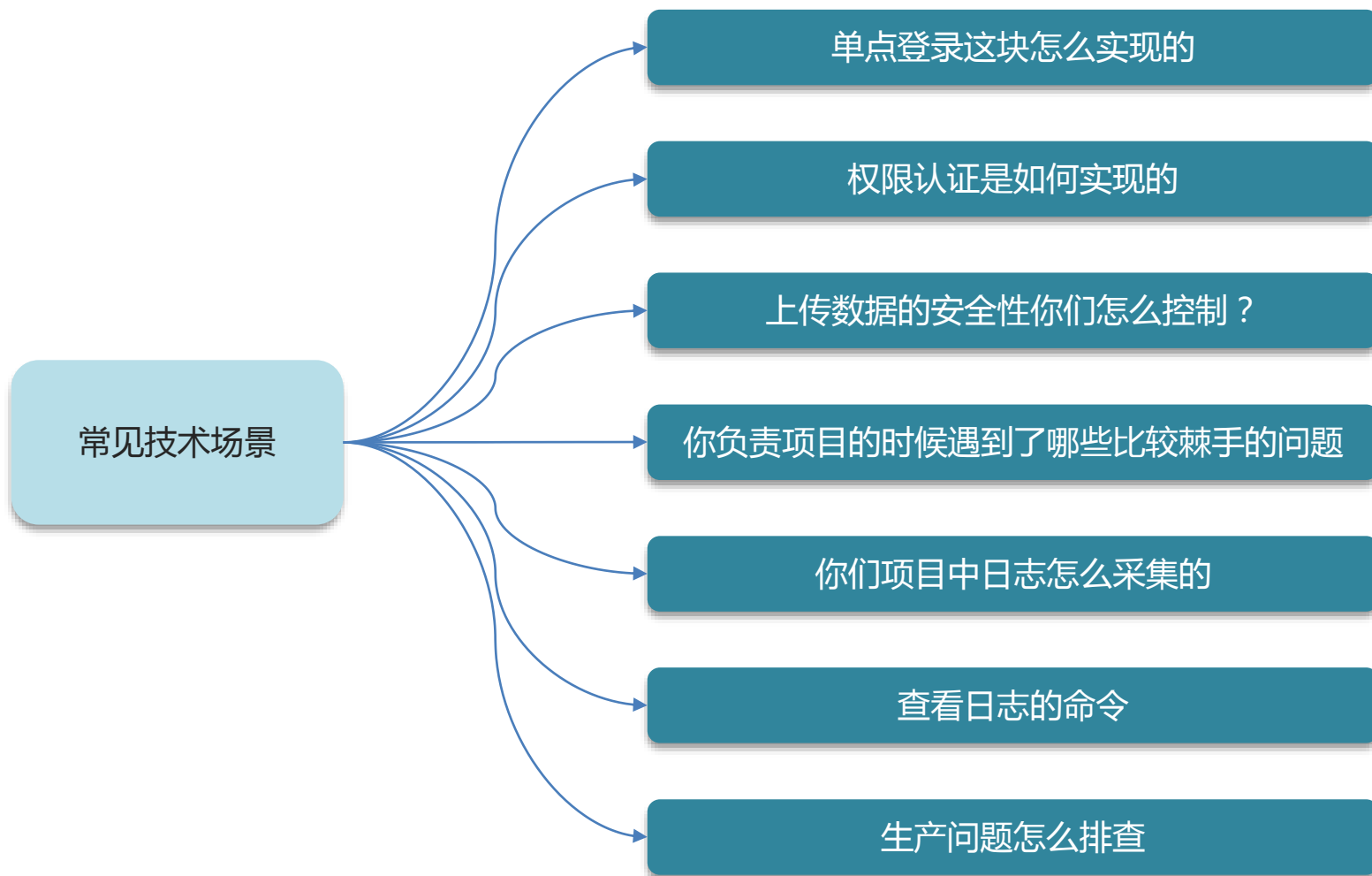


常见技术场景



黑马程序员
www.itheima.com

传智教育旗下
高端IT教育品牌





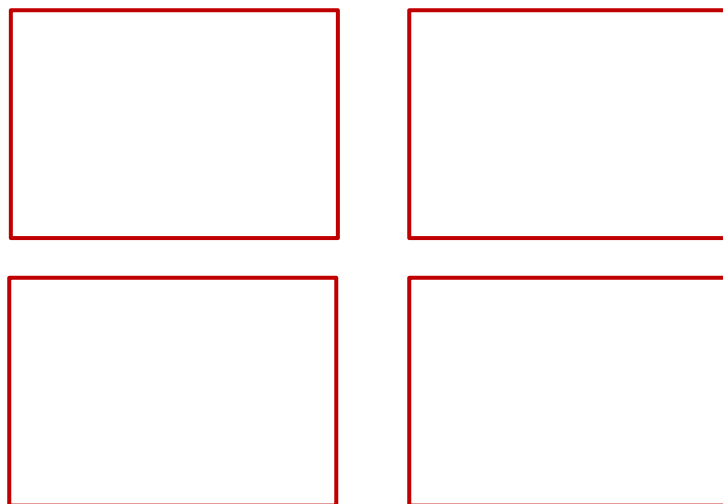
单点登录这块怎么实现的

单点登录的英文名叫做：Single Sign On (简称**SSO**) ,只需要登录一次，就可以访问所有信任的应用系统



单体

单个tomcat服务session可以共享



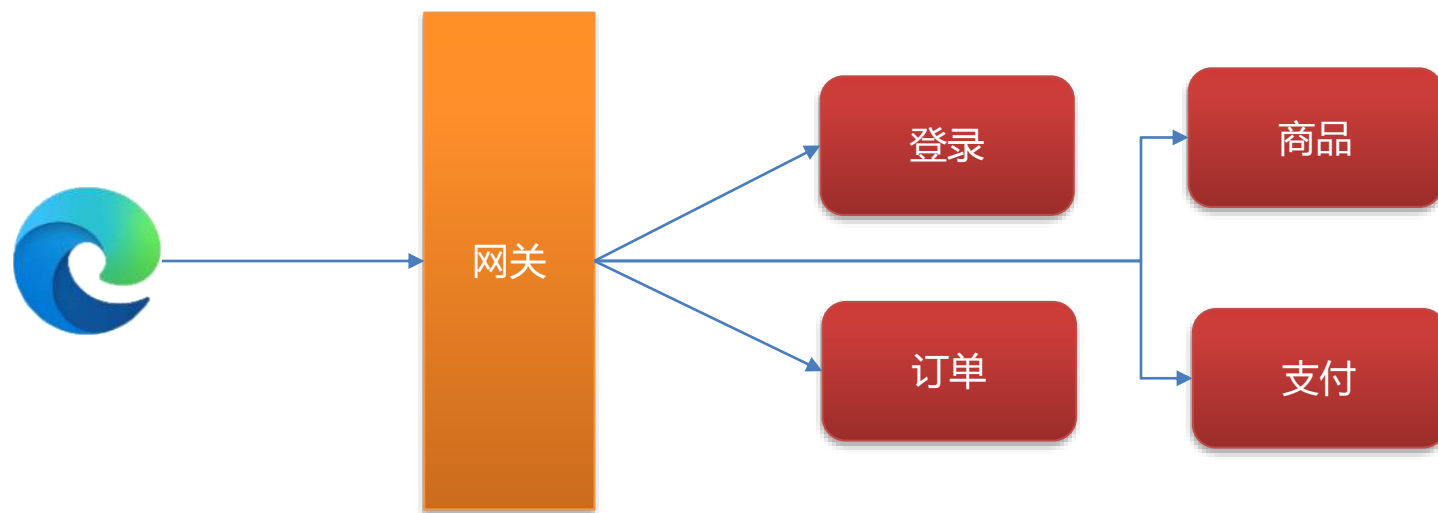
微服务、分布式

多个tomcat服务session不共享

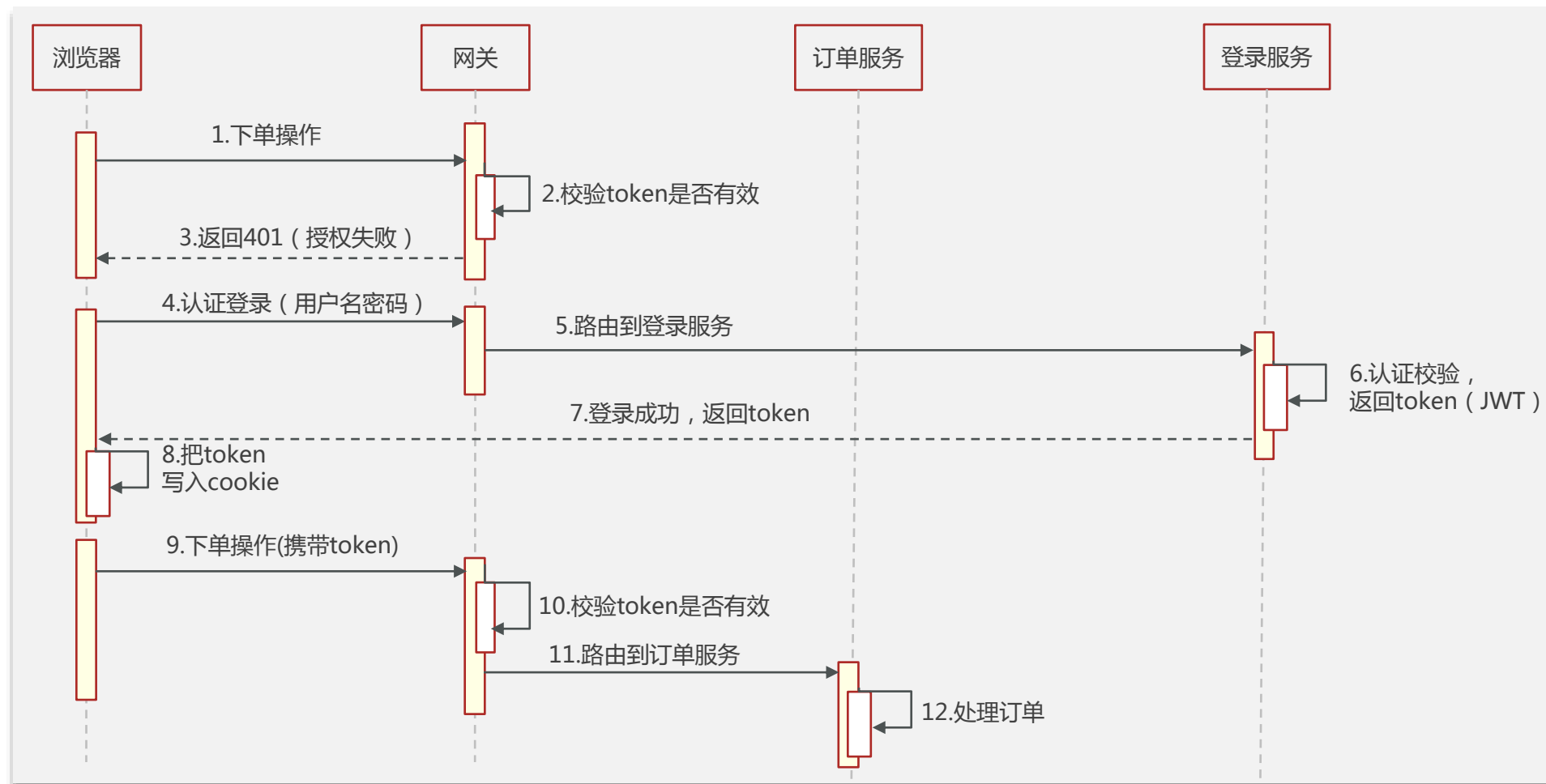
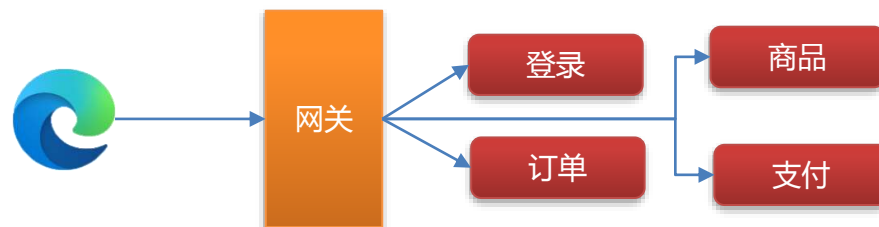
单点登录解决方案

- JWT(常见)
- Oauth2
- CAS
- ...

JWT解决单点登录



JWT解决单点登录





单点登录这块怎么实现的

1，先解释什么是单点登录

单点登录的英文名叫做：Single Sign On（简称**SSO**）

2，介绍自己项目中涉及到的单点登录（即使没涉及过，也可以说实现的思路）

3，介绍单点登录的解决方案，以JWT为例

- I. 用户访问其他系统，会在网关判断token是否有效
- II. 如果token无效则会返回401（认证失败）前端跳转到登录页面
- III. 用户发送登录请求，返回浏览器一个token，浏览器把token保存到cookie
- IV. 再去访问其他服务的时候，都需要携带token，由网关统一验证后路由到目标服务



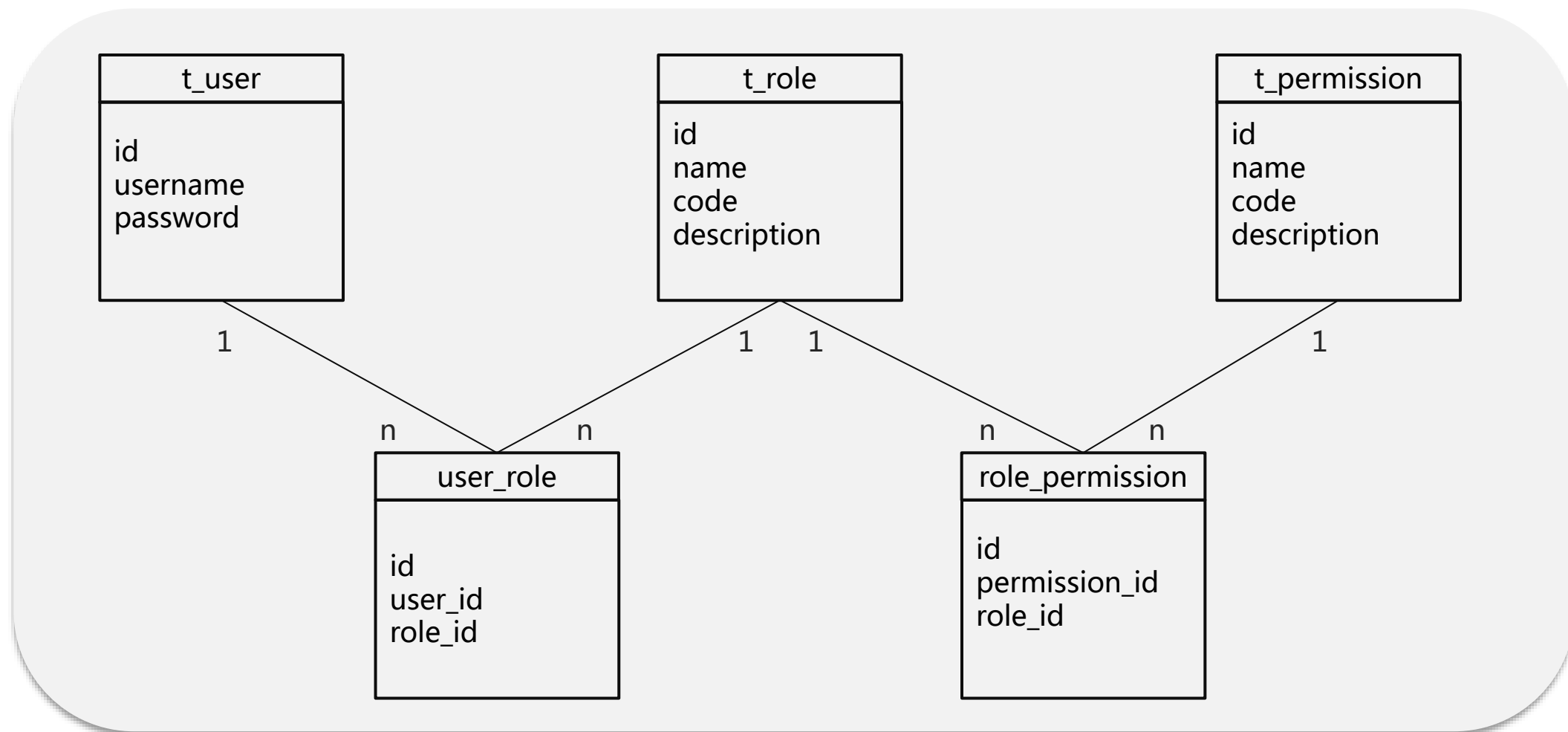
权限认证是如何实现的

后台的管理系统，更注重权限控制，最常见的就是RBAC模型来指导实现权限

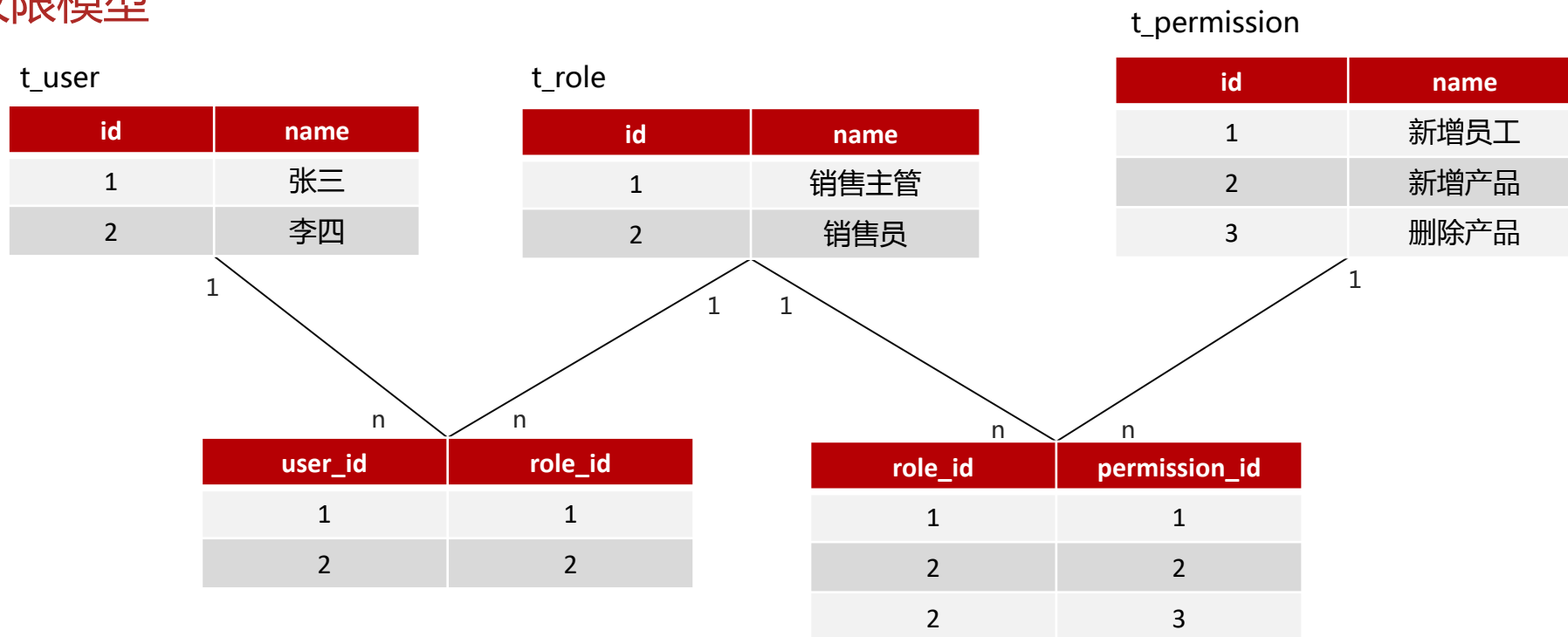
RBAC(Role-Based Access Control)基于角色的访问控制

- 3个基础部分组成：用户、角色、权限
- 具体实现
 - ◆ 5张表（用户表、角色表、权限表、用户角色中间表、角色权限中间表）
 - ◆ 7张表（用户表、角色表、权限表、菜单表、用户角色中间表、角色权限中间表、权限菜单中间表）

RBAC权限模型



RBAC权限模型



张三具有什么权限呢？

流程：张三登录系统---> 查询张三拥有的角色列表--->再根据角色查询拥有的权限

权限框架：

Apache shiro 、 Spring security (推荐)

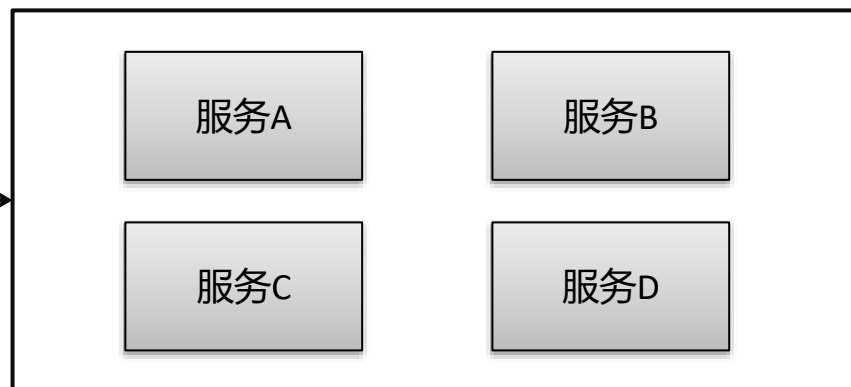


权限认证是如何实现的

- 后台管理系统的开发经验
- 介绍RBAC权限模型5张表的关系（用户、角色、权限）
- 权限框架：Spring security



上传数据的安全性你们怎么控制？

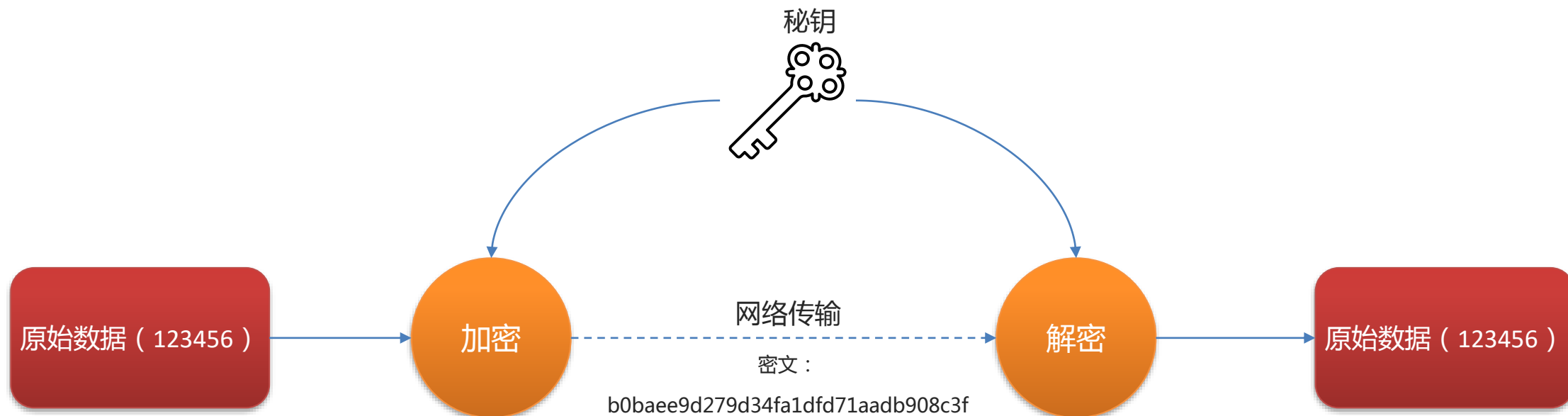


后台服务

使用非对称加密（或对称加密），给前端一个公钥让他把数据加密后传到后台，后台负责解密后处理数据

对称加密

文件加密和解密使用相同的密钥，即加密密钥也可以用作解密密钥



优点：加密速度快，效率高

缺点：相对不太安全（不要保存敏感信息）

非对称加密

两个密钥：公开密钥（publickey）和私有密钥，公有密钥加密，私有密钥解密

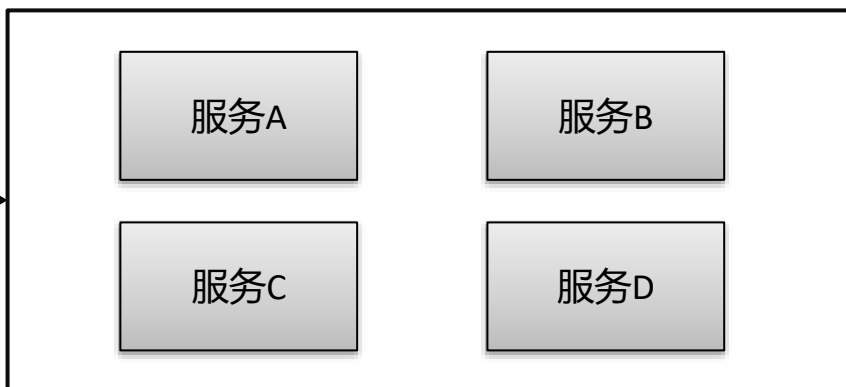


优点：与对称加密相比，安全性更高

缺点：加密和解密速度慢，建议少量数据加密



上传数据的安全性你们怎么控制？



后台服务

使用**非对称加密**（或**对称加密**），给前端一个公钥让他把数据加密后传到后台，后台负责解密后处理数据

- 文件很大建议使用对称加密，不过不能保存敏感信息
- 文件较小，要求安全性高，建议采用非对称加密



你负责项目的时候遇到了哪些比较棘手的问题？怎么解决的

提前准备

01 设计模式

工厂

策略

责任链

...

02 线上BUG

CPU飙高

内存泄漏

线程死锁

...

03 调优

慢接口

慢SQL

缓存方案

...

04 组件封装

分布式锁

接口幂等

分布式事务

支付通用

- 1, 什么背景（技术问题）
- 2, 过程（解决问题的过程）
- 3, 最终落地方案



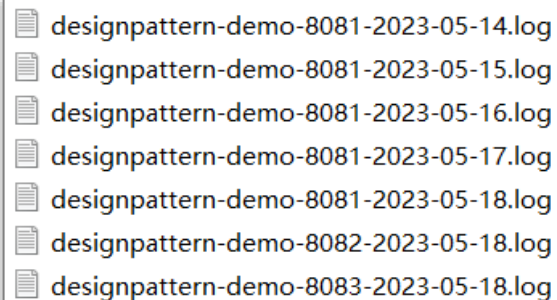
你们项目中日志怎么采集的

1，为什么要采集日志？

日志是定位系统问题的重要手段，可以根据日志信息快速定位系统中的问题

2，采集日志的方式有哪些？

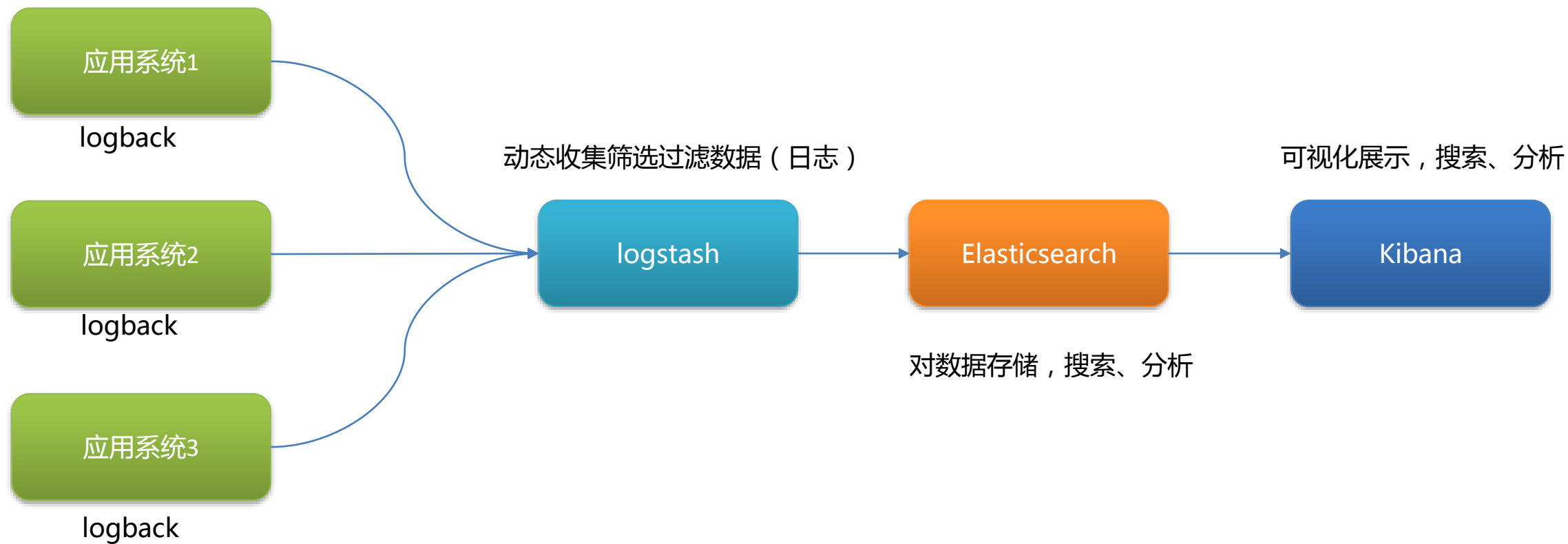
- ELK：即Elasticsearch、Logstash和Kibana三个软件的首字母
- 常规采集：按天保存到一个日志文件



```
designpattern-demo-8081-2023-05-14.log  
designpattern-demo-8081-2023-05-15.log  
designpattern-demo-8081-2023-05-16.log  
designpattern-demo-8081-2023-05-17.log  
designpattern-demo-8081-2023-05-18.log  
designpattern-demo-8082-2023-05-18.log  
designpattern-demo-8083-2023-05-18.log
```


你们项目中日志怎么采集的

ELK即Elasticsearch、Logstash和Kibana三个开源软件的缩写





你们项目中日志怎么采集的

- 我们搭建了ELK日志采集系统
- 介绍ELK的三个组件：
 - Elasticsearch是全文搜索分析引擎，可以对数据存储、搜索、分析
 - Logstash是一个数据收集引擎，可以动态收集数据，可以对数据进行过滤、分析，将数据存储到指定的位置
 - Kibana是一个数据分析和可视化平台，配合Elasticsearch对数据进行搜索，分析，图表化展示



查看日志的命令

目前采集日志的方式：按天保存到一个日志文件

Linux中查看日志：

- 实时监控日志的变化
- 按照行号查询
- 按照关键字找日志的信息
- 按照日期查询
- 日志太多，处理方

```
designpattern-demo-8081-2023-05-14.log
designpattern-demo-8081-2023-05-15.log
designpattern-demo-8081-2023-05-16.log
designpattern-demo-8081-2023-05-17.log
designpattern-demo-8081-2023-05-18.log
designpattern-demo-8082-2023-05-18.log
designpattern-demo-8083-2023-05-18.log
```

```
<!-- 定义日志文件的存储地址, 使用绝对路径 -->
<property name="LOG_HOME" value="/home/logs"/>
<!-- 按照每天生成日志文件 -->
<appender name="FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
  <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
    <!-- 日志文件输出的文件名 -->
    <fileNamePattern>${LOG_HOME}/${springAppName}-${serverPort}-${yyyy-MM-dd}.log</fileNamePattern>
  </rollingPolicy>
  <encoder>
    <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} [%thread] %-5level %logger{36} - %msg%n</pattern>
  </encoder>
</appender>
```

logback配置

查看日志的命令

- 实时监控日志的变化

实时监控某一个日志文件的变化：`tail -f xx.log`；实时监控日志最后100行日志：`tail -n 100 -f xx.log`

- 按照行号查询

- ◆ 查询日志尾部最后100行日志：`tail -n 100 xx.log`

- ◆ 查询日志头部开始100行日志：`head -n 100 xx.log`

- ◆ 查询某一个日志行号区间：`cat -n xx.log | tail -n +100 | head -n 100` (查询100行至200行的日志)

- 按照关键字找日志的信息

查询日志文件中包含debug的日志行号：`cat -n xx.log | grep "debug"`

- 按照日期查询

`sed -n '/2023-05-18 14:22:31.070/,/ 2023-05-18 14:27:14.158/p' xx.log`

- 日志太多，处理方式

- ◆ 分页查询日志信息：`cat -n xx.log | grep "debug" | more`

- ◆ 筛选过滤以后，输出到一个文件：`cat -n xx.log | grep "debug" > debug.txt`

```
2023-05-18 14:27:13.416 [main] INFO o.s.b.w.e.tomcat.Tomc
2023-05-18 14:27:13.425 [main] INFO o.a.catalina.core.Sta
2023-05-18 14:27:13.426 [main] INFO o.a.catalina.core.Sta
2023-05-18 14:27:13.552 [main] INFO o.a.c.c.C.[Tomcat].[1
2023-05-18 14:27:13.552 [main] INFO o.s.b.w.s.c.ServletWe
pleted in 1417 ms
2023-05-18 14:27:13.761 [main] INFO o.s.s.c.ThreadPoolTas
2023-05-18 14:27:14.150 [main] INFO o.s.b.w.e.tomcat.Tomc
2023-05-18 14:27:14.158 [main] INFO com.itheima.UserAppli
```



生产问题怎么排查

已经上线的bug排查的思路：

- 1，先分析日志，通常在业务中都会有日志的记录，或者查看系统日志，或者查看日志文件，然后定位问题
- 2，远程debug(通常公司的正式环境（生产环境）是不允许远程debug的。一般远程debug都是公司的测试环境，方便调试代码)

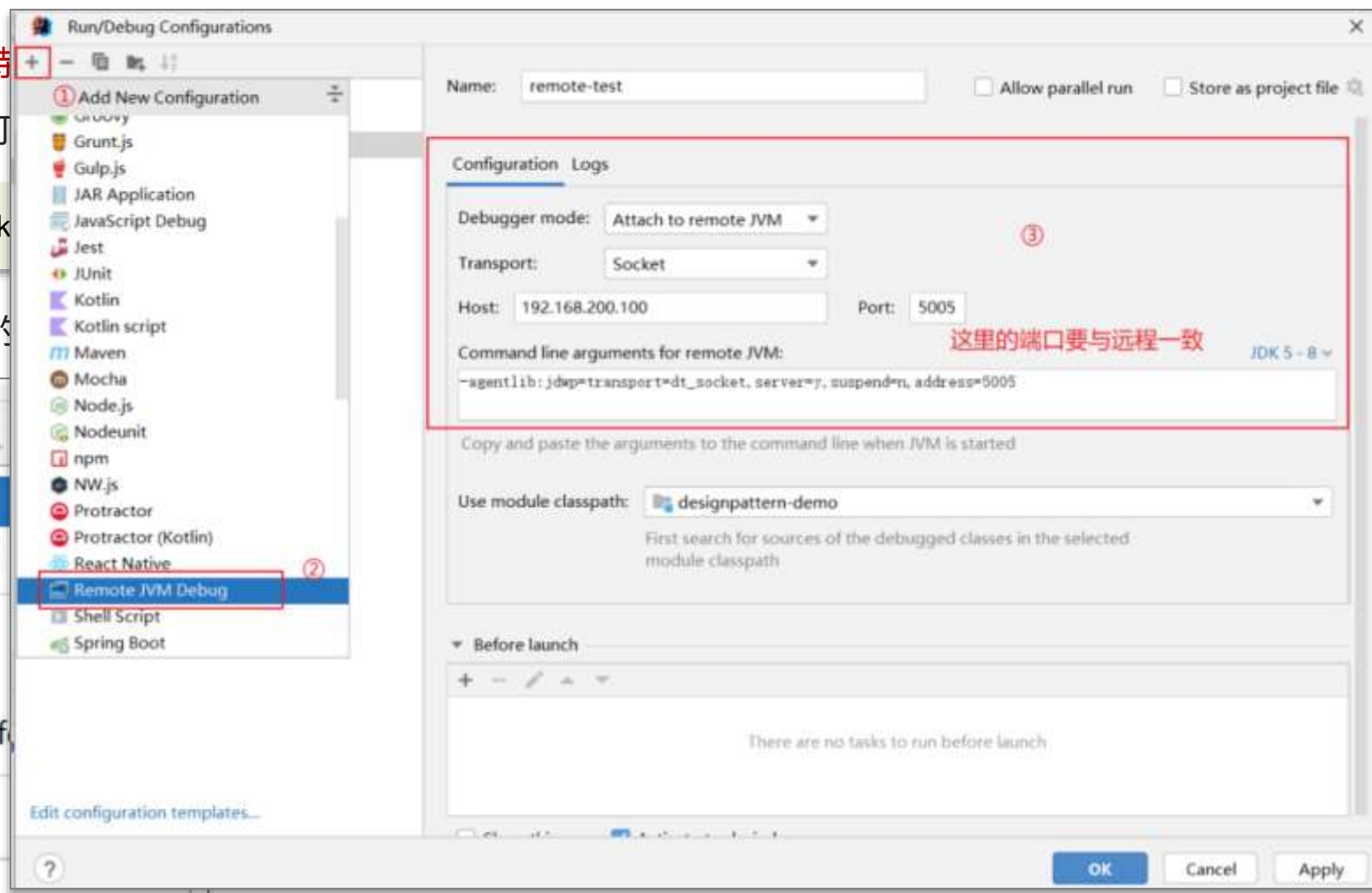
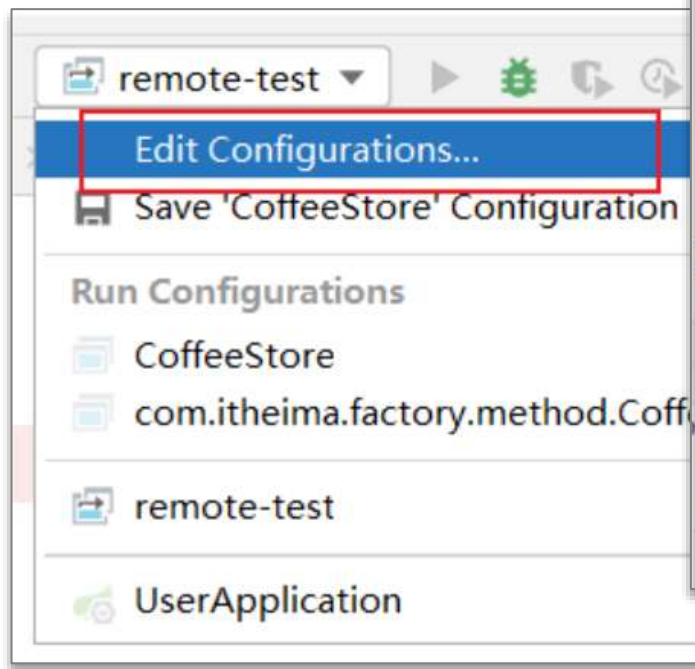
远程debug

前提条件：远程的代码和本地的代码要保持

1. 远程代码需要配置启动参数，把项目打

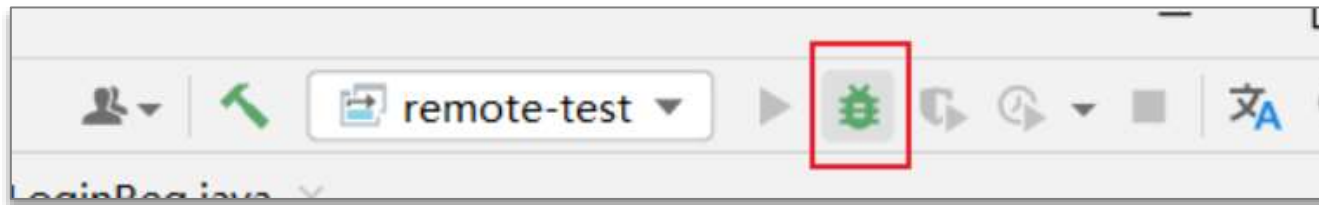
```
java -jar -agentlib:jdwp=transport=dt_socket
```

2. idea中设置远程debug，找到idea中的



远程debug

3. idea中启动远程debug



4. 访问远程服务器，在本地代码中打断点即可调试远程



生产问题怎么排查

已经上线的bug排查的思路：

- 1，先分析日志，通常在业务中都会有日志的记录，或者查看系统日志，或者查看日志文件，然后定位问题
- 2，远程debug(通常公司的正式环境（生产环境）是不允许远程debug的。一般远程debug都是公司的测试环境，方便调试代码)



怎么快速定位系统的瓶颈

- 压测（性能测试），项目上线之前测评系统的压力
- 监控工具、链路追踪工具，项目上线之后监控
- 线上诊断工具Arthas(阿尔萨斯)，项目上线之后监控、排查





传智教育旗下高端IT教育品牌