

信用算方
www.xysl.com

从单体到微服务再到中台战略的历程



2019年12月15日

信用算力首席架构师

十余年互联网架构经验

精通Microservice Architecture , 大数据

拥有亿级用户平台架构经验

万级并发的API网关经验。



潘志伟



目录

01 什么是中台

02 基础服务治理到底怎么做

03 共享服务的诞生

04 中台为快反而生



/// 什么是中台

迷雾中

最前面的那辆车
如何开？



定义：企业级能力复用平台

- 企业级：中台服务的范围
- 能力：中台所承载的对象，如业务能力，数据能力
- 复用：中台的核心价值，各业务系统所要求能力复用
- 平台：中台的主要表现形式

必要性

- 是刚需吗？
- 有多个不同的业务线需要支持吗？
- 存在重复建设吗？

时机

- 符合公司战略发展方向
- 高层领导的全力支持
- 业务部门协同支持配合
- 基础服务治理是否就绪



基础服务治理到底怎么做？

为什么辛苦搭建的微服务没办法上线？

案例：

由于时间紧迫，团队一边做着需求迭代以及新功能上线，一边还得做新架构开发，每个人都非常的忙碌。然而，在大家辛苦半个月后第一个版本交付测试的时候却出现了问题，每个人负责的模块的代码风格不统一，框架结构不统一，其他人如果想参与到对应的模块却不清楚从哪里入手，且培训成本过高。

准备微服务工具

直接生成PO DTO对象

自动生成pom依赖

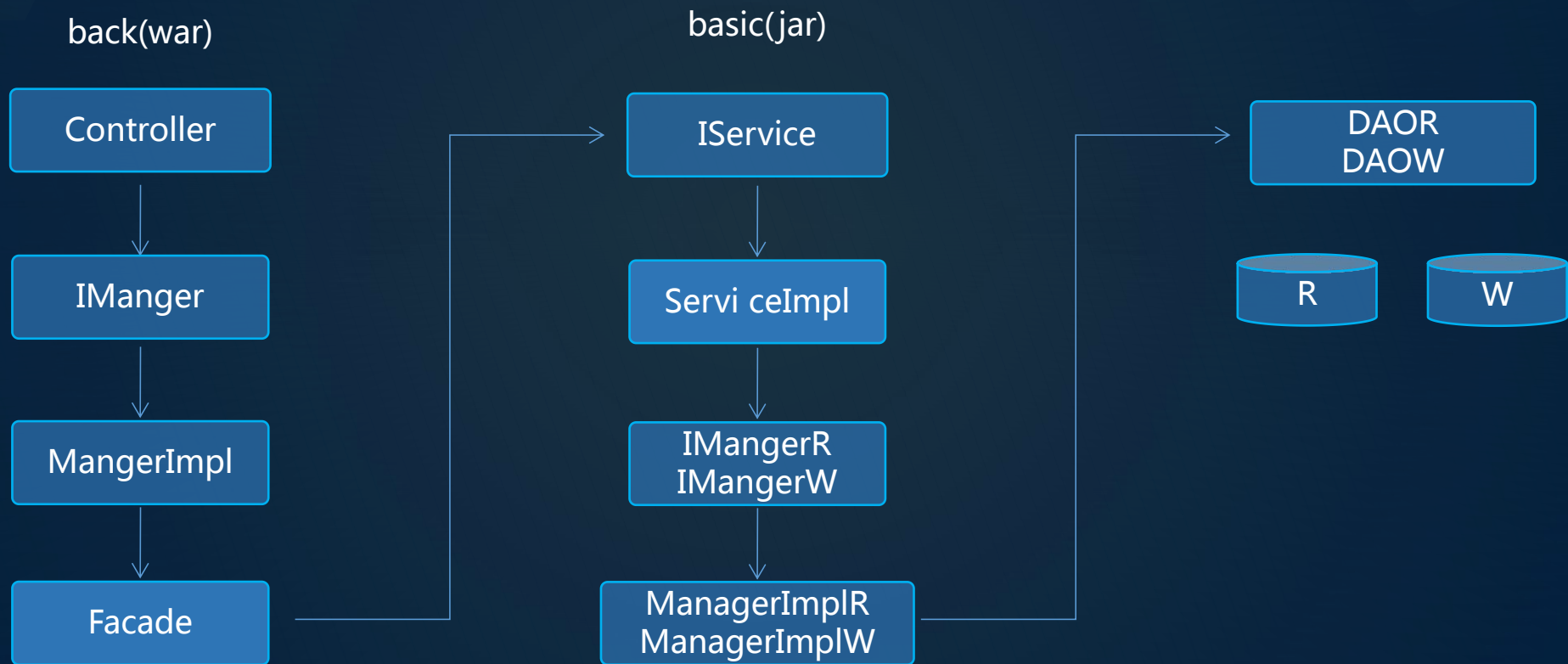
自动生成代码骨架

自动生成读写分离

自动生成Mybatis xml文件



生成后的代码结构以及依赖关系



代码分层

Back (业务逻辑层)
back-project

business: 业务处理层, 调用过多个服务在这里做聚合
facade: 外部服务调用统一出口
model: 定义VO相关的字段
web: 接收网关转过来的请求

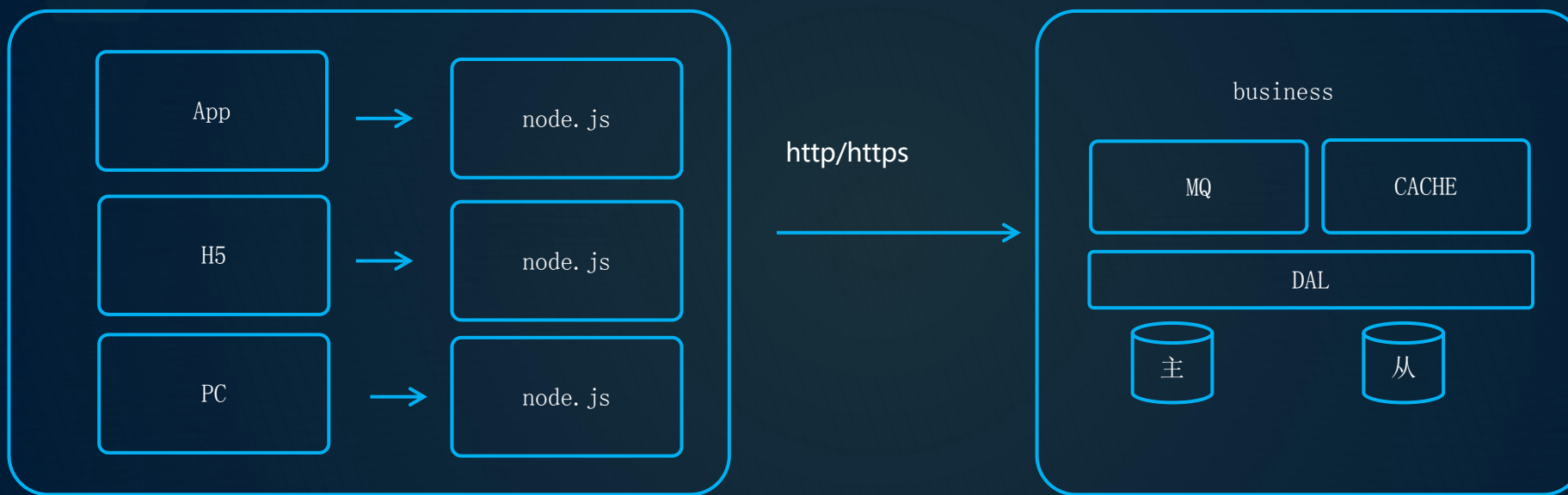
Basic (原子服务层)
basic-project

api: 接口层, 外部服务依赖该接口提供的服务;
service: 接口实现层, 实现api接口
business: 数据相关的处理; 以及PO对象
model: 定义DTO对象
facade: 基础服务调用外部服务

Basic (原子服务) 层关注点

- 单表原则，禁止2张以上的表做join查询
- 不要包含任何业务逻辑
- 对外屏蔽分库分表的操作
- 根据数据量的大小引入KV类型的分布式存储，如HBase等
- 如需要跨库跨表需要引入检索工具，如ES等

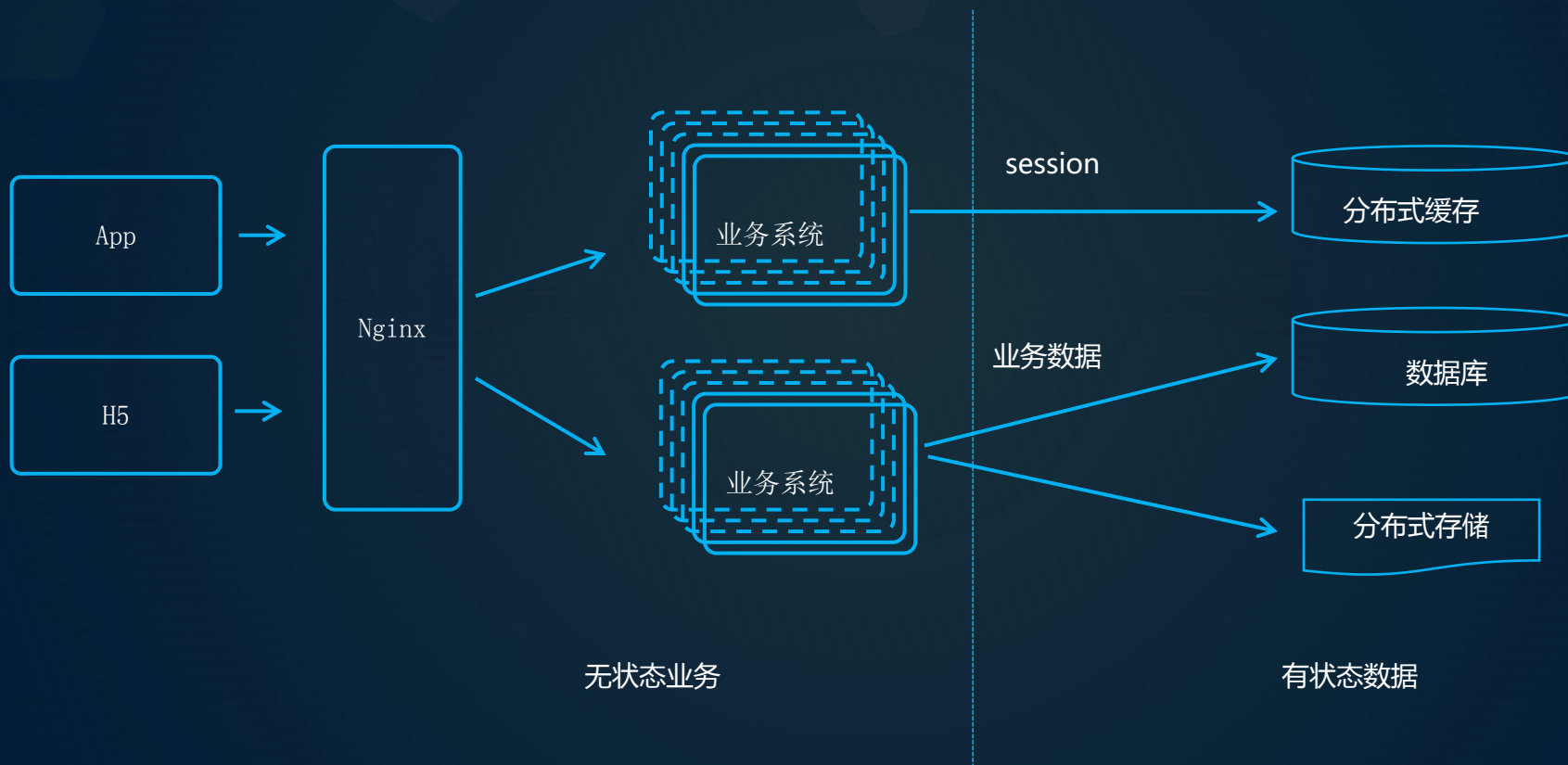
前后端分离

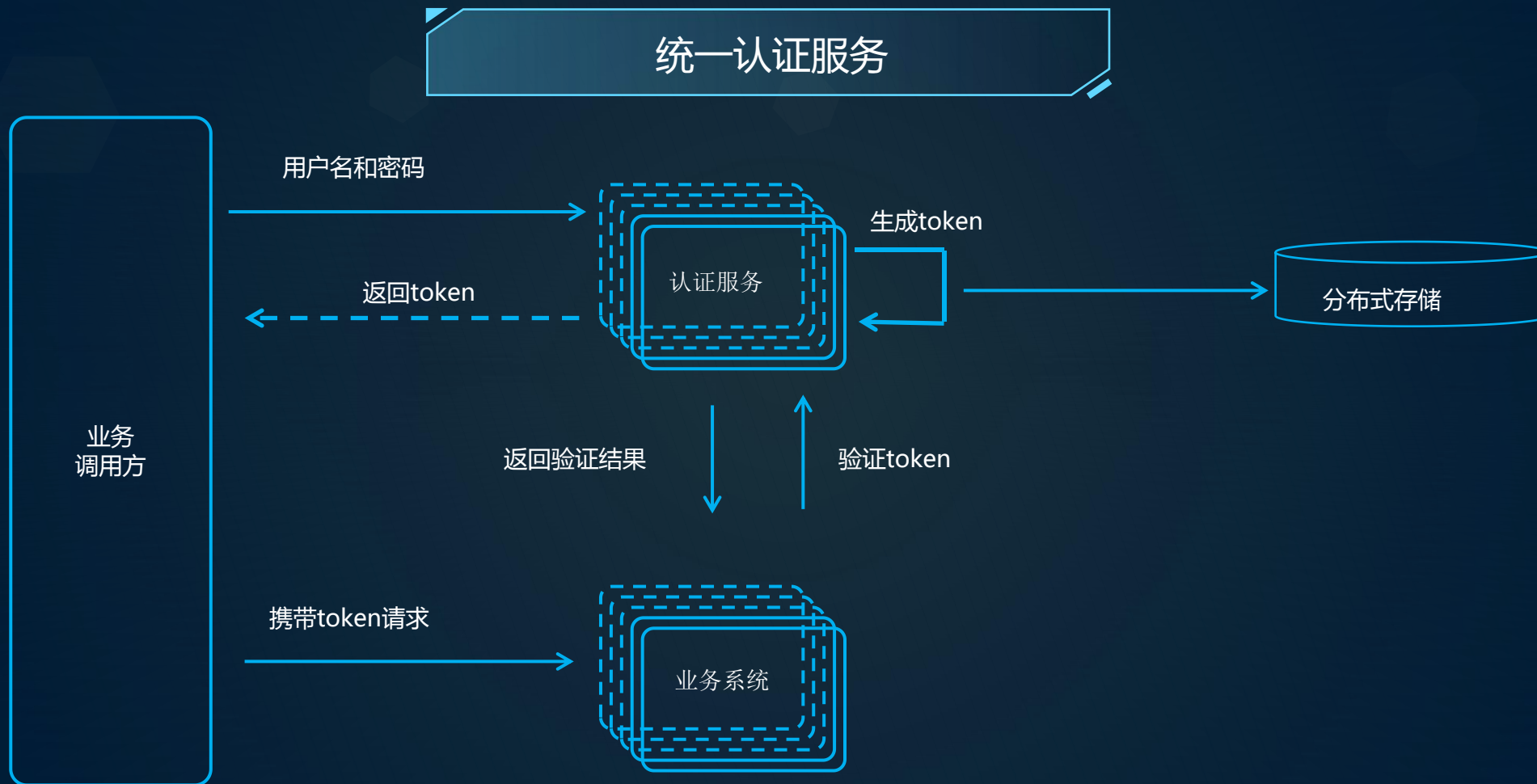


前端开发人员

服务器端开发人员

所有服务无状态化





工具总结

- ✓ 代码未编工具先行
- ✓ 统一微服务工程结构
- ✓ 统一服务启动方式（jar/war）
- ✓ 统一缓存调用方式（架构封装统一提供jar包和底层存储无关）
- ✓ 统一MQ调用方式（架构封装统一提供jar，和具体MQ类型无关）
- ✓ 统一日志格式
- ✓ 统一多服务依赖调用方式(串行调用方式、并行调用方式)
- ✓ 统一熔断、降级处理流程

业务架构图（简化版）





千人千面 个性化推荐

App

H5

用户服务

产品服务

属性服务

校验服务

标签服务

订单服务

排序服务

用户行为

黑镜服务

精准营销

版本服务

.....

化串行为并行，提升访问速度



熔断功能的要求

熔



断

% 错误率

人工干预

🕒 时间窗口

📅 31 主动告警

降级目的：业务高峰的时候，去掉非核心链路，保证主流程正常运行

熔断目的：防止应用程序不断地尝试可能超时或失败的服务，能达到应用程序执行而不需要等待下游修正服务

推荐：hystrix 或 Sentinel

快、再快、更快



用户服务

产品服务

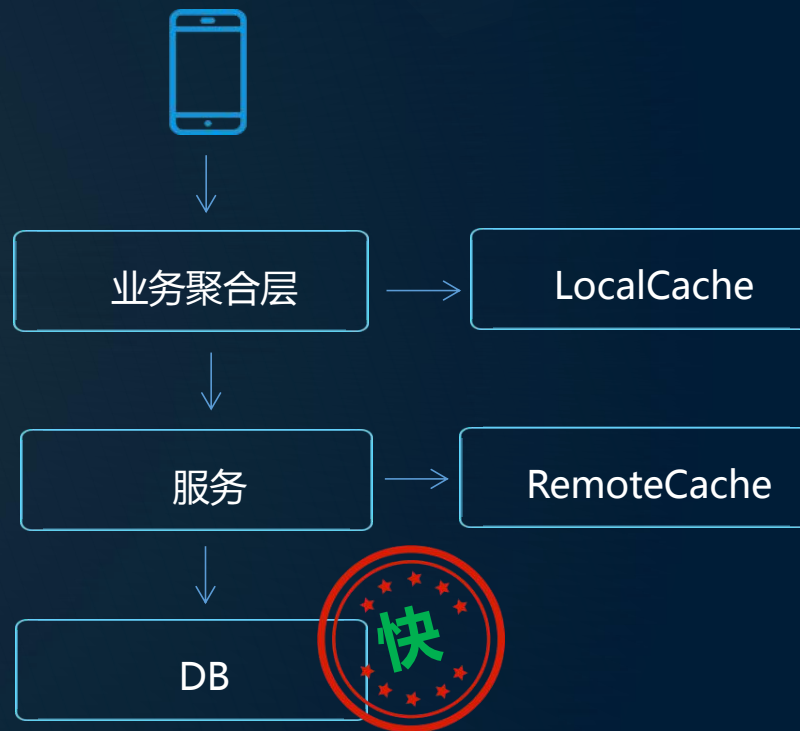
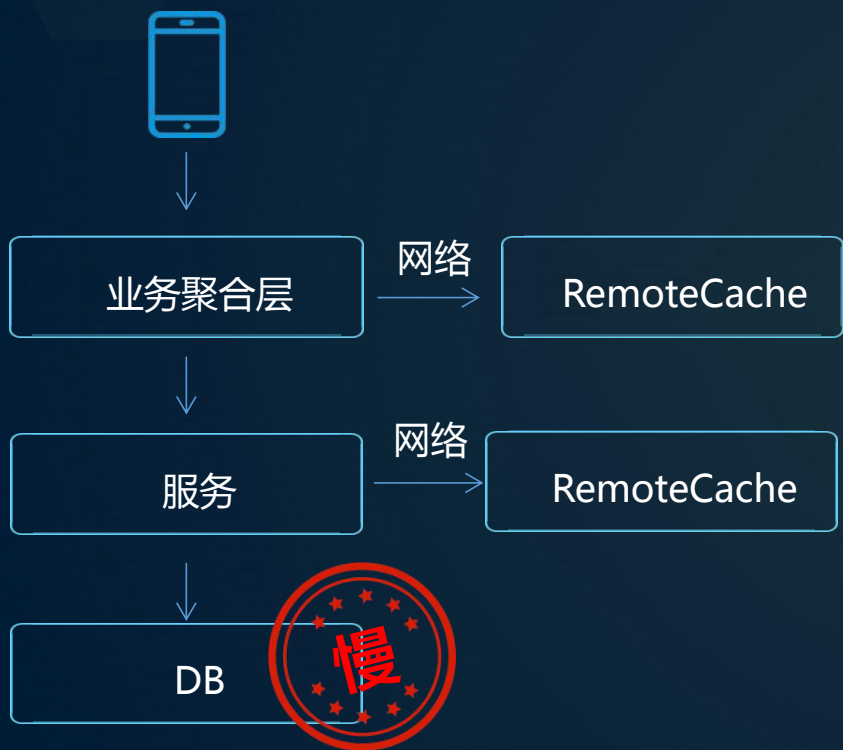
属性服务

校验服务

标签服务

精准营销

充分使用缓存



缓存总结

统一接口

本地缓存

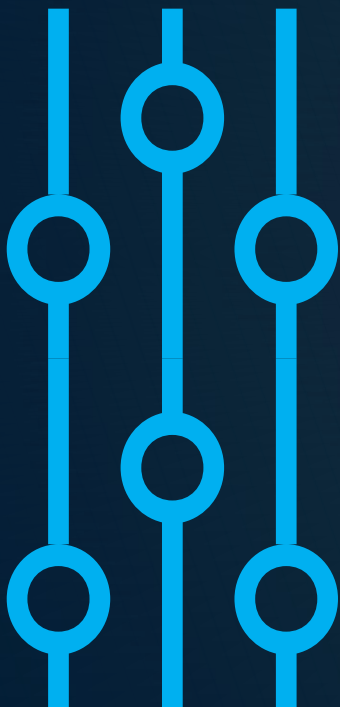
布隆过滤器

分布式缓存

- ✓本地缓存：Caffeine、Guava 更新策略：TTL自动过期
- ✓分布式缓存：redis
- ✓缓存穿透：伪造数据库中不存在的值，导致缓存命中失败，直接查询DB -> BloomFilter
- ✓缓存击穿：缓存key刚好过期，大量同样的key请求过来，导致直接命中数据库 -> 查询数据层使用互斥锁
- ✓缓存雪崩：同一个时刻缓存大面积失效，请求全部查询DB -> 服务限流+告警



基于MQ的应用解耦



- 应用层必须支持消息幂等
- 支持消息回溯
- 支持消息重放
- 基于关键字查询
- 消息的消费的机器IP以及消息时间

架构迁移阶段需求迭代怎么做

场景：正在忙着做服务化的改造，需求迭代来了怎么办？

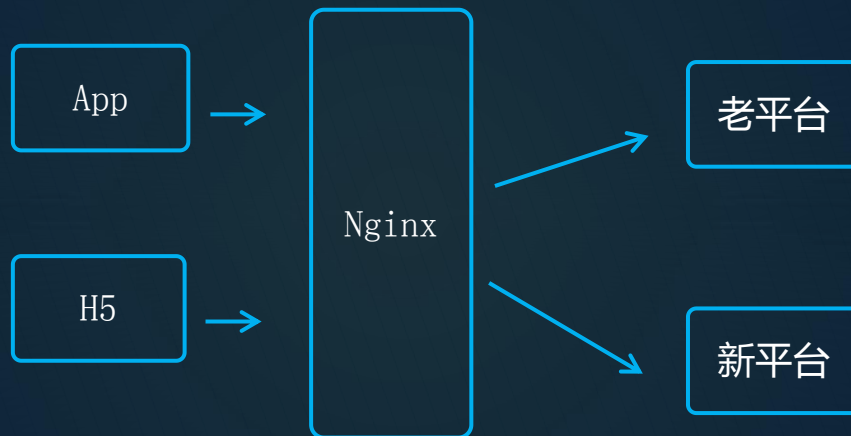
推荐

- 新功能新架构
- 历史功能变更也在新架构上开发

杜绝

- 在老的工程上直接做服务化的改动
- 在老的工程上继续做需求迭代

新老系统线上如何兼容运行



- Nginx 配置路由 20% -> 50% -> 100%
- 新老平台接口名称一致、参数一致、结果格式一致、序列化方式一致

服务治理总结

- ✓ 多个服务调用尽可能并行化调用；
- ✓ 本地缓存+远程缓存完美搭配，提供统一调用方式
- ✓ 服务高可用熔断降级必不可少，但是参数配置需要清晰
- ✓ MQ的解耦和消峰功能是微服务有效搭配



/// 共享服务的诞生



乘风

智能营销平台

整合全域流量资源，提供实时风控前置的精准获客服务，帮助金融机构提升营销ROI。



巡风

智能风控系统

数据、技术、策略、模型，四位一体集成输出，以图形化配置工具实现风险自主管理、策略自主部署、业务自主拓展。



御风

智能运营平台

涵盖审批、交易侦测、客服、调额、催收等环节，保障零售信贷快捷化、规模化、精细化运营需求。



产品线

- 大部分功能重叠
- 形成内部小圈子不共享

系统架构

- 自己造的轮子跑的最快
- 我的代码永远是优秀的，其他都是渣渣

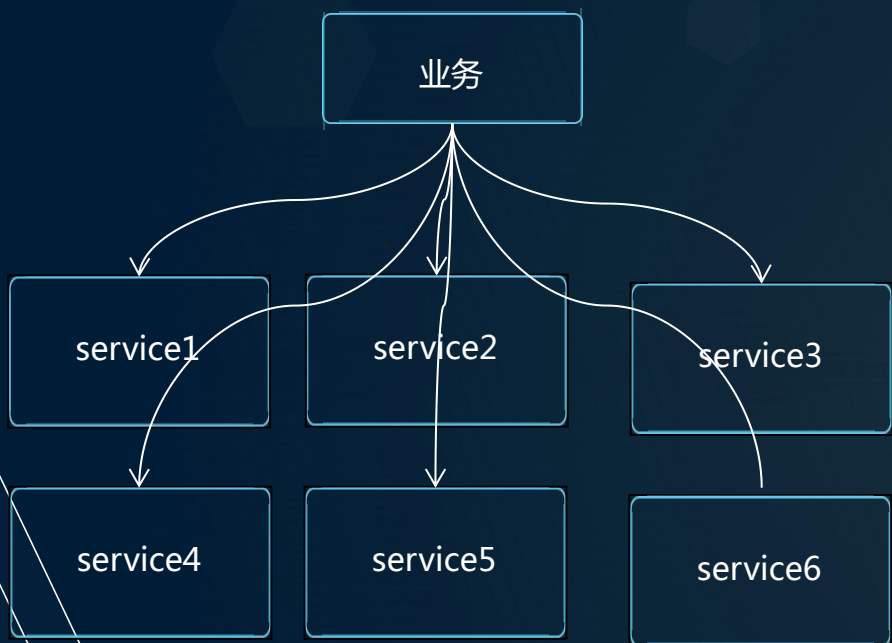
数据架构

- 每个系统的数据相互独立
- 统计口径由产品线来制定

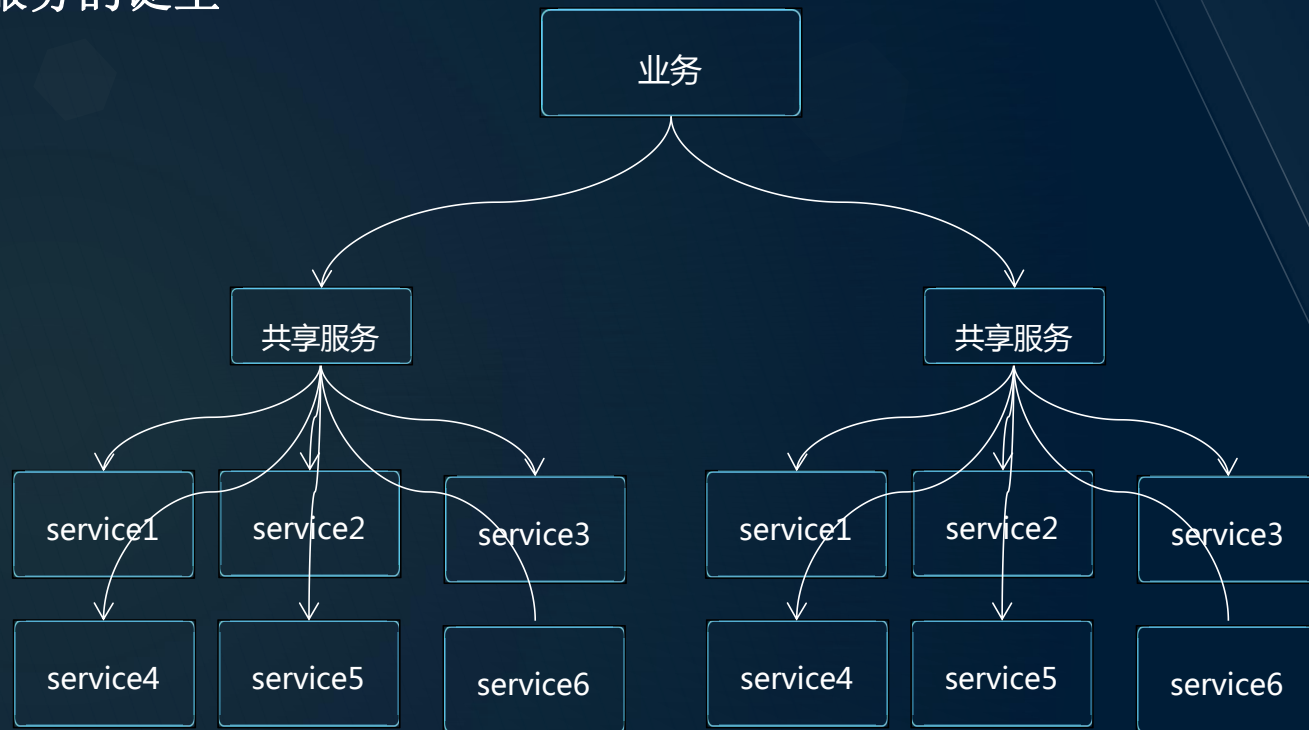
组织架构

- 部门之间资源不共享
- 资源不够，永远缺人

共享服务的诞生



- ✓业务侧需要知道明确的依赖关系
- ✓业务侧需要知道明确的调用关系
- ✓业务侧需要控制好降级、熔断
- ✓一旦业务侧换人又需要重新熟悉



- ✓共享服务承担所有核心业务逻辑
- ✓共享服务控制降级，熔断
- ✓业务侧只需要调用单一接口



中台为快反而生

中台为快反而生

快

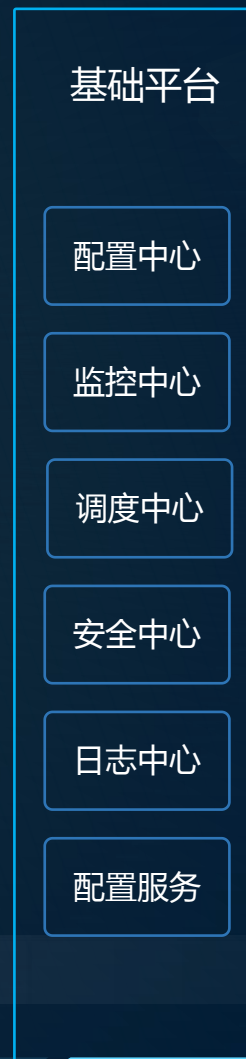
- ✓快速迭代,朝令夕改
- ✓3-5人2周完成创新项目

反

- ✓降级试错成本
- ✓快点失败、早点失败、小点失败

中台组织新架构



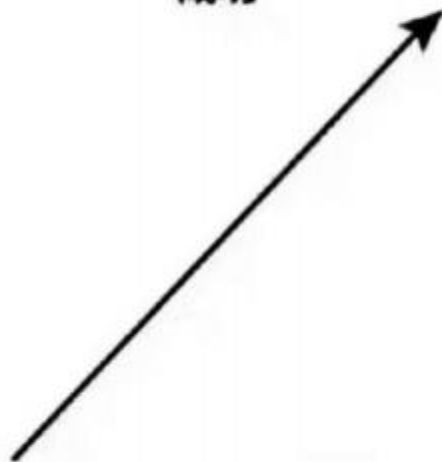


总结：

- ✓不是所有项目都需要中台
- ✓基础服务、必须稳定可靠
- ✓确定好各自的边界范围
- ✓制定明确的KPI指标
- ✓业务高度从项目中抽象
- ✓中台必须要高层介入，最好是最高领导，全员共识
- ✓中台是演变出来的，而不是一次性做出来的

SUCCESS

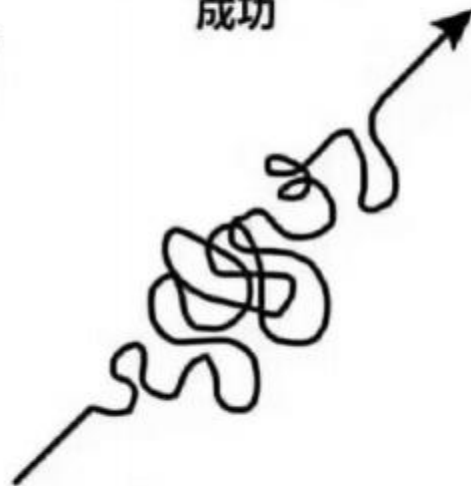
成功



what people think
it looks like
很多人认为


SUCCESS

成功



what it really
looks like
其实成功是这个样子



潘志伟 

上海 浦东新区



扫一扫上面的二维码图案，加我微信



THANKS
ALL