学相伴旅游项目实战 - **Pug**后台管理系统

> 编撰人：徐柯
> 编撰时间：*2021*年*12*月*25*日，圣诞节
> 编撰地点：万达星巴克和狂神碰面

项目阶段

- 阶段一：单体项目开发和上线
- 阶段二：从单体到高可用集群演进
- 阶段三：逐个击破分布式核心的问题以及实战开发
- 阶段四：微服务项目的迭代和部署发布
- 阶段五：容器化部署Docker和K8s容器化
- 阶段六：性能优化阶段

01、项目的架构和技术栈

架构模式：

前后端分离的架构开发模式。

后端技术

开发工具：IntelliJ IDEA2021、Maven3.6.3、MySQL5.7、Tomcat9、Git / Gitee

后端技术：SpringBoot2.6.2(SpringMVC5、Spring5、Mybatis3)、SpringCloud/AlibabaCloud、阿里云OSS、SMS(阿里云短信)、Quartz、POI、SpringSecurity、Redis、Jwt

IDEA	**重点快捷键、插件、基本设置**
Maven3.6.3	重点九大概念、模块化开发
MySQL5.7	重点SQL语句、表设计
Tomcat9	掌握安装、使用、核心配置
Linux(CentOS7)	掌握安装、网络配置NTA、相关命令、Java运行环境搭建
Redis6	重点5种数据类型、基本命令、Jedis、乐观锁、事务
SpringMVC5	重点MVC设计模式、Controller层代码开发、数据接收与响应、注解
Spring5	重点IOC和AOP、声明式事务
Mybatis3	重点DAO映射、动态SQL、连表SQL语句
SpringCloud/AlibabaCloud	掌握基本原理及使用
Ehcache	本地缓存，解决缓存击穿的问题
OSS(阿里云)	存储图片
SMS(阿里云短信)	重点发手机验证码
Quartz	重点Cron表达式
Vue3.x.js	掌握常用指令、MVVM模型、axios异步请求
ElementUI	掌握常用组件、根据官网案例能设计简单页面
POI	掌握Excel读取、写入
SpringSecurity	掌握RBAC、认证与授权、密码加密
百度地图	掌握API基本使用
Echarts	掌握饼图、折线图、柱状图
uniapp	掌握app开发的技巧
Nginx	集群部署
SharedingJdbc/MyCAT	实现分库和分表，独写分离
Seata	实现分布式事务
Docker/K8s	容器化部署

前端技术

后台：自研，参考 若依

前端技术：Vue3.x.js + ElementUI + Node.js + npm+ Echarts+自研组件

02、项目的整体架构如下：

03、项目介绍

""游啦啦 ""旅游是一款应用于旅游管理机构的业务系统，主要提供跟团游、散拼团、小包团、自由行、自助游、游轮旅游、自驾游、定制游以及景点门票预订、机票预订、火车票预订服务，还有牛人专线、首付出发旅游等品质高端、价格实惠的旅游路线。该系统主要分为两个方向，一个是后台管理人员使用的PC端后台系统，一个是用户进行旅游订阅的移动端。后台管理系统的主要功能模块有：基础权限控制、自由行、跟团游、小包团、酒店、散拼团、旅拍等和统计功能等功能。前台系统有注册登录、套餐列表展示、套餐详情展示、旅游预约、旅游预约下单和预约成功信息展示等功能。

- 携程
- 马蜂窝
- 穷游
- 去哪儿
- 途牛

04、项目架构的概述

企业尤其是发展到一定规模以后，往往会面临业务种类繁多，业务高度依赖的问题。而随着业务的发展，企业内的部门也会越来越多，分工也越来越细，部门之间的依存度和沟通成本也会越来越高。

如果企业缺乏总体的规划，缺乏应对时长变化的快速反应能力和高效支撑商业模式创新的机制，企业的运营和创新的成本也将会大大的增加。为了提高时长响应能力，解决商业模式的创新问题，越来越多的企业开始尝试数字化转型。

这个过程中，企业项目的架构也会随着波动和变化。

05、传统企业的数字化转型过程

随着企业的发展和自主研发的能力和提升，企业信息系统建设可能会经历产品外购，外包，自主研发，甚至集团统一运营建设阶段。

- 第一个阶段：以产品购买为主。这个阶段企业的规模较小，或者是初创阶段，短时间内难以形成自主研发的能力，一般采用外购产品和系统。存在的问题就是：早期购买成熟的套件产品，往往是会把很多的功能打包在一个软件系统中，也称之为单体应用，即“一个系统打天下”的局面。导致产品的耦合度超高，扩展能力不强，出现很多用不着，不能灵活定制的等问题。
- 第二阶段：自主研发为辅，外包为主：随着企业业务的发展，企业的技术人员得到一定的补充，并具备了研发和设计的能力以后，企业系统建设模式会慢慢转型为以自主研发为辅，外包为主的模式。存在的问题就是：产品的自主权不能自己控制，拖延扯皮，沟通就会变得非常的复杂。
- 第三个阶段：自主研发为主，外包为辅：当企业内部研发人员达到相当规模，企业IT团队有能力实现核心业务全流程自动研发，现在大部分的企业都会选择这种开发模式。好处就是：企业能够完全的自控和可控。
- 第四个阶段：集团统一运营模式：对于大型跨国经营的集团，不同子公司之间可能由于早期缺少统一规划，因此在技术栈和应用系统上存在较大的差异，导致集团内部出现大量的IT重复投入和重复建设，难以在集团内实现交叉销售和商业模式的创新。

  集团的统一运营模式是通过集团内统一规划，统筹管理，统一运营，重构中台业务模型、统一技术标准、统一云环境和优化运营体系等，实现集团内各子公司之间业务流程的共享和联通，技术体系的标准统一和资源共享信息系统建设模式。

  近些年来，越来越多的大型企业开始学习阿里巴巴的，实施中台数字化转型战略。这种模式改变了集团内各子公司“山头林立，各自为政”的IT建设方式，可以在整个集团实施企业级业务能力复用和集团统一运营。

06、服务技术架构的演进

系统应用架构的变迁

01、原始分布式时代

可能与绝大多数人的认知有些差异，“使用多个独立的分布式服务共建一个更大型的系统”的设想与实际的尝试，其实要比今天大家所了解的大型单体系统出现的时间更早。

在上个世纪70年代末80年代初，计算机科学刚经历了从以大型机为主向微型机为主的蜕变过程。此时的微型计算机系统具有16位寻址能力，不足5MHZ时钟频率的处理器和128kb左右的内存地址空间。当时的计算机硬件有限的运算能力和处理能哪里，已直接影响到了单台计算机上信息系统软件能够达到的最大规模。这个阶段分布式架构是最原始的探索，从结果来看，历史局限决定了它不可能一蹴而就地完成分布式的难题。但是从过程来看，这个阶段的探索称得上成绩斐然。研究的过程中很多成果都对今天计算机科学的诸多领域产生了深远的影响。并直接推动了后续软件架构的演化进程。

02、单体架构时代

单体架构系统是今天绝大多数软件开发者都学习，实践过的一种软件架构，在整个软件架构演进的历史进程这里，出现最早，应用范围最广，使用人数最多，统治历史最长的一种架构风格。这个阶段通常采用面向过程的设计方法，系统包括客户端的UI层和数据库层两层，通常采用C/S架构，大多采用结构化编程方式，系统围绕数据库驱动设计和开发，总是从设计数据库和字段开始。

Web应用程序发展的早期，大部分web工程是将所有的功能模块(service side)打包到一起并放在一个web容器中运行，很多企业的Java应用程序打包为war包。其他语言(Ruby，Python或者C++)写的程序也有类似的问题。假设你正在构建一个在线商店系统：客户下订单、核对清单和信用卡额度，并将货物运输给客户。很快，你们团队一定能构造出如下图所示的系统。

一个归档包（例如**war**格式或者**Jar**格式）包含了应用所有功能的应用程序，我们通常称之为单体应用。架构单体应用的方法论，我们称之为单体应用架构，这是一种比较传统的架构风格。

01、单体架构的缺陷

> 一句话： 集中式的单体架构往往会将多个功能放到一个应用中，经过日积月累，这个应用就会编程一个庞大而复杂的“怪物”， 随着项目团队成员的更替，时间一长就很少有人能够完全的搞懂这些代码之间的逻辑关系。有些人可能会因为担心修改遗留代码而出现不可预知的*BUG*，，而宁愿增加大量不必要的代码，这样会导致应用越来越庞大，越来越复杂，可读性越来越差，最终陷入恶性循环。对于整个团队来说，系统研发工作编程了一件及其痛苦的事情。

优点	缺点
容易测试，在本地就可以启动完整的系统	需求的增加，不断往容器中添加服务，显得臃肿笨拙
容易部署，直接打包为一个完整的包，拷贝到容器中即可运行	修改一个地方则需要整个系统重新部署
容易开发：开发方式简单，方便运行和调试（适用于较为简单的系统）	不利于更新技术框架，除非将系统重构
.	维护成本大，扩展性差

03、SOA开发架构时代

SOA是`Service-Oriented Architecture`的英文缩写，就是面向服务的架构。这里的服务可以理解为**service**层业务服务。

企业服务总线（**ESB**）：

`ESB是面向服务架构(SOA)的核心构成部分`，指传统数据连接技术（web、xml、中间件技术）结合的产物，简单来说，就是一根管道，用来连接各个服务节点，为了集成不同系统，不同协议的服务，服务总线做了消息的转化解释和路由工作，让不同的服务互联互通；是一个具有标准接口、实现了互连、通信、服务路由。

> 总线需要具备的功能：
>
> *1*、服务统一管理：为整个系统提供一个统一、标准。可靠 、可扩展的服务管理平台。
> *2*、集成服务：提供基础的服务与定制的服务，支持集成服务模式，支持服务的分解，服务调度和路由、封装及组合。
> *3*、公用服务：提供内置的各种公共服务，如，认证服务、日志服务等。
> *4*、服务协议转换：通过把不同的通信协议转换为标准的报文，屏蔽异构系统的底层技术差异。
> *5*、服务监控：提供服务等级管理及流量管理。
> *6*、安全体系：提供多种安全机制并支持和第三方安全系统的有效集成。

SOA是集成多个较大组件（一般是应用）的一种机制，将整体构成一个彼此协作的套件，一般来说，每个组件会从始至终执行一个完整的业务逻辑。SOA中包含多个服务，服务之间通过相互依赖最终提供一系列的功能。一个服务通常以独立的形式存在于操作系统进程中。各个服务之间，通过网络调用。

特点：

1、系统集成：从系统角度讲，解决了企业系统与系统间通信问题，把原来散乱、无规划的系统间的网状结构梳理成规整，可治理的系统。在梳理时则需要引用一些产品，`常用的是企业服务总线（ESB）`、技术规范、服务管理规范。`主要解决核心问题，无序变有序`。

2、系统的服务化：从功能角度讲，把业务转换成可复用、可组装的服务，通过服务的编排实现业务的快速复制。目的是把原先固有的业务功能转变为通用的业务服务，实现快速复用。主要解决的核心问题，原来固有业务可复用。

3、业务的服务化：从企业的角度讲，把原来职能化的企业架构转变为服务化的企业架构，进一步提升企业的对外服务能力。把一个业务单元封装成一项服务。主要解决的核心问题是高效。

优点	缺点
数据统一，共享数据库，使服务接口使用同一的数据模型的数据，确保数据一致性	技术不匹配，在某些情况并不能轻松对操作平台进行重新打包，原因是业务功能结构需求不匹配
灵活性较高，缩短产品和服务的上线时间，降低了开发与改变流程的成本	系统间交互需要使用远程通讯，一定程度上降低了响应速度
系统由子系统组成，系统易于重构	.

04、微服务架构开发时代

跟SOA类似，在SOA上做了升华，微服务架构强调业务需要彻底的组件化和服务化，在微服务架构中，系统的业务逻辑被拆分成为一系列小而松散耦合的分布式组件（组件一般指应用），共同构成较大的应用。每个组件都被称为微服务，而每个微服务都在整体架构中执行着单独的任务或单独的功能。每个微服务可能会被一个或多个其它微服务调用，以执行较大应用需要完成的具体任务。

比如：假设一个APP中有积分体系功能，现需要更新积分，只需要更新发布积分的微服务即可，其他的功能正常运行，不受影响。而不是将整个业务系统都停掉，所有的功能都要暂停一会儿，等发布积分的服务才能使用。

特点：

1、通过服务实现组件化，开发者不再需要协调其它服务部署对本服务的影响。

2、按业务能力来划分服务和开发团队，开发者可以自由选择开发技术，提供API服务 即可。

3、去中心化:

- 每个微服务都有自己私有的数据库来持久化业务数据。
- 每个微服务只能访问自己的数据库，而不能访问其他服务的数据库。
- 某些业务场景下，需要在一个事务中更新多个数据库，这种情况也不能直接访问其它微服务的数据库。
- 降低微服务之间的耦合度，不同服务可以采用不同的数据库技术。在复杂的的业务场景下，如果包含多个微服务，通常在客户端或者中间层(网关)处理。

优点	缺点
部署简单，每个服务承担少数职责，波及范围小	系统整体延迟增加，原来的函数调用改为服务调用
易于扩展，某一项服务的性能达到瓶颈，只需增加该服务的节点数即可，其它服务不改变	每个服务都需要单独部署，运维、测试成本增加
减低资源的耦合性，服务独立，数据源唯一	前期服务的定义和拆分需要较大工作量
易于维护，每个微服务的职责单一，复杂性降低，不会牵一发而动全身	.

面向服务架构**(SOA)**与微服务的主要区别：

功能	面向服务**(SOA)**架构	微服务架构
耦合性	一般是松耦合	总是松耦合
公司架构	任何类型	小型、专注于功能交叉团队
管理	着重中央管理	着重分散管理
目标	确保应用能够交互操作	执行新功能、快速拓展开发团队
系统组成	由多个子系统组成	由多个组件组成
应用部署	相互依赖，部署复杂	独立部署，互不影响
数据管理	全局数据模型，共享数据库	每个服务都有自己的数据模型或数据库
服务架构	企业服务总线，集中式的服务架构	无集中式总线，松散的服务架构

05、中台

中台并非完全是自上向下的战略设计，也非是为了追随行业风口，可以理解为一种产品设计思路或系统架构的思路。

中台是随着公司业务高速发展，组织不断膨胀的过程中暴露的问题需要解决。将企业的核心能力随着业务不断发展以数字化形式沉淀到平台，形成以服务为中心，由业务中台和数据中台构建起数据闭环运转的运营体系，供企业更高效的进行业务探索和创新。

中台做到前后分离，后台统一提供数据接口，前台实现业务流转。

中台与微服务的区别：

中台是提升企业的能力的复用，一种方法论/思想。
微服务是独立开发、维护、部署的小型业务组件，一种技术架构。

中台与微服务的关系：

微服务架构，是实现中台思想的落地的重要手段。

中台解决的核心问题：

为减少重复业务系统开发及实现系统数据共享一个技术平台底座，将多年技术沉淀的价值最大化，统一各个业务部门或系统重复使用、重复建设的功能和系统统一规划和管理。

什么时候需要中台：

如阿里：淘宝，有订单、库存、评价、积分、物流等业务系统。天猫也有订单、库存、评价、积分、物流等业务系统。1688，也有类似业务系统。多个系统有重复业务系统需要建设，且系统间数据不能完全共享，系统各自运行。此时使用技术中台以及业务中台，来实现业务重用及数据共享，把技术沉淀价值最大化。

> 借用网上的回复：
> 微服务就是：将整个军队分散为若干军区，每个军区之间确定各自驻防的边界划分，至于各军区如何行军、如何存放军火、如何部署兵力，不作统一规定，各军区自行决断，各军区的人当然也不能进入其他军区的地盘，但各作战单位必须遵守共同的通讯频道，必须满足对其他军区的服务契约。
> 中台就是：建立强大的火箭军、炮兵。空军。无人机部队、信息化部队等，如此一来，前方作战小分队可以很小，一个班的人要攻打一个山头，只要侦查清楚这个山头的特定地形和敌军布防，然后就根据情况呼叫空军地毯式轰炸。呼叫炮兵火力覆盖、呼叫无人机定点清除、呼叫信息化部队电子干扰...一个班就可以搞定。
> 虽然这两个概念并不互斥，但是微服务听起来更像是【守城】，就是对现有地盘的加强和巩固。而中台更支持【开拓】，主要目的是更灵活的拓展新的业务。

07、大型网站架构的演进

01、最初形态 (静态网页)

02、Javaweb的时代

这个时候数据开始可以进行维护和交互，单体架构的时代来临。

03、单体架构的模式

最初形态

分离文件服务器和数据库服务器

可以缓解应用服务器的压力。

错误的思维：

不是你买的服务越多，做了集群，你网站就抗住高并发。因为如果服务器如果都不在一个局域网内的，服务器与服务器通讯是通过`公网IP`通讯， 通过带宽访问，如果你每个服务器的带宽1M

正确的思维：

- 要么自己买服务器，安装虚拟机，自己分配每个服务器内存的大小。每个服务器已进程运行在一个电脑中。这速度肯定非常快的。
- 要么就买局域网，这个时候你需要和阿里云协商。

缓存中间件形态

这个时候数据库的问题和压力越来越大，引入缓存来缓解压力。

04、集群架构的模式

读写分离版本

读写分离版本数据库集群版本

搜索引擎技术

05、微服务架构方式

异步消息队列方式