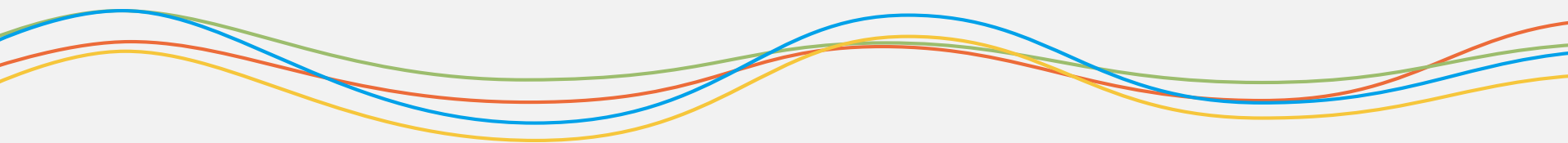



# 从业务架构到微服务

张辉清 | 中青易游 CTO



 微服务的调用应该放哪一层？

- A. 表现层
- B. 业务逻辑层
- C. 数据层
- D. 公共层

- A. 业务架构
- B. 领域模型
- C. 接口设计
- D. 程序方法

答：

企业总体架构：企业服务=》业务架构=》领域模型=》面向服务架构=》微服务架构

单个项目架构：功能需求=》用例图=》用例活动图=》领域模型=》代码实现

- 微服务与企业总体架构
- 微服务与单个项目架构
- 微服务技术实现

# 微服务与企业总体架构

## ● 业务架构

## ● 架构现状

- 功能架构
- 应用架构
- 数据设计
- 物理架构
- 接口架构
- 领域模型

## ● 架构规划

- 顶层架构规划
- 功能规划
- ...

## ● 架构实施

- ▷ 1 系统概述
- ▲ 2 企业商务模型
  - 2.1 商务模式
  - 2.2 商务主体
  - 2.3 主营业务
  - 2.4 组织结构
  - 2.5 商务运作模型
- ▷ 2.6 国内机票业务流程
- ▷ 2.7 国际机票业务流程
- 2.8 附档资料
- ▷ 3 信息系统模型
- ▷ 4 应用架构
- ▷ 5 数据设计
- ▷ 6 物理架构
- ▷ 7 接口架构
- ▷ 8 领域模型
- ▷ 9 架构规划
- ▲ 10 架构实施
  - 10.1 整体思路
  - 10.2 Roadmap
  - 10.3 改造一期
  - 10.4 改造二期
  - 10.5 改造三期

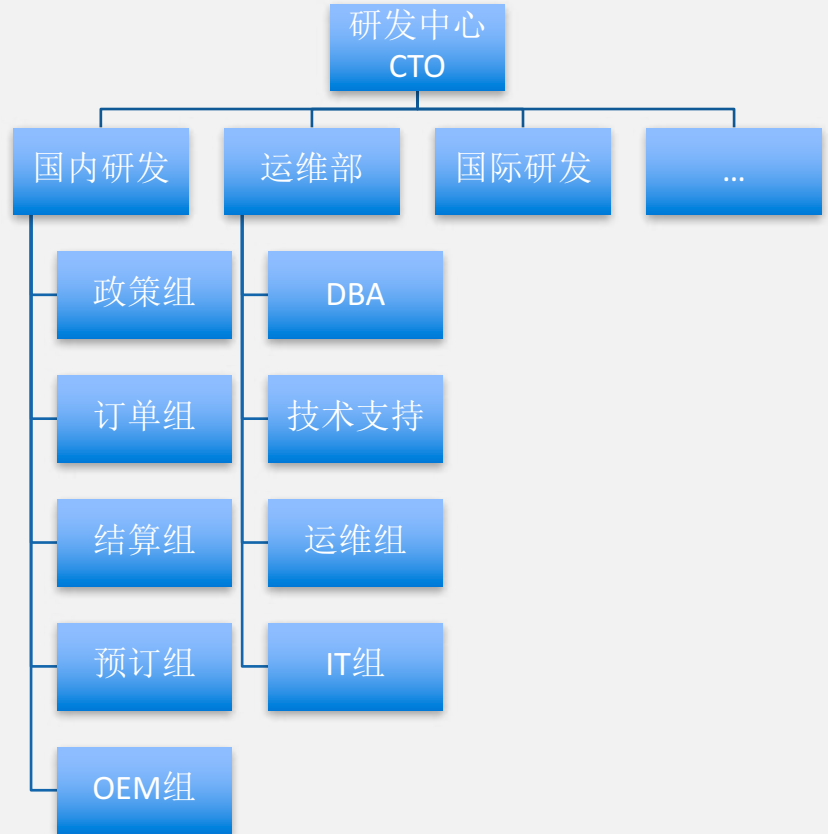
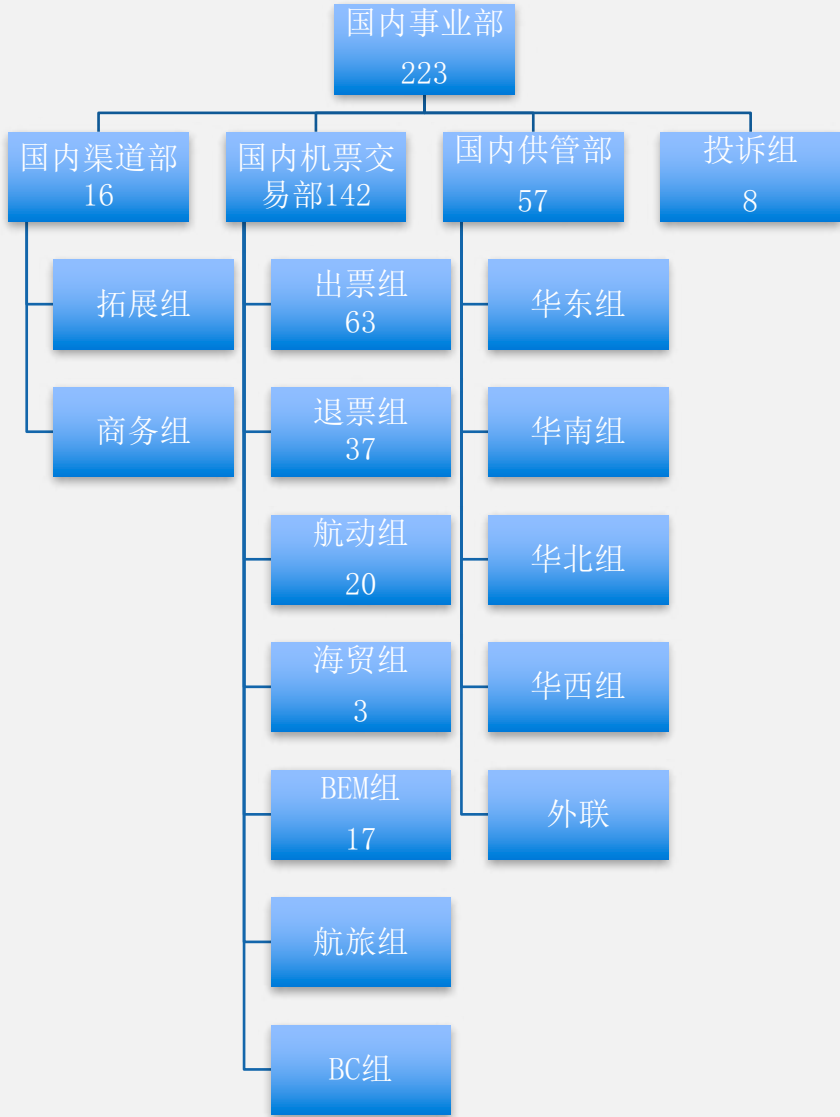
- ▲ 3 信息系统模型
  - 3.1 国内全部功能
  - 3.2 国内角色
  - 3.3 国内角色与功能
  - 3.4 国际全部功能
  - 3.5 国际角色
  - 3.6 国际角色与功能
- ▷ 4 应用架构
- ▲ 5 数据设计
  - 5.1 国内数据库
  - 5.2 国内数据表
  - 5.3 国内E-R图
  - 5.4 国内订单状态
  - 5.5 国际数据库
  - 5.6 国际数据表
  - 5.7 国际E-R图
  - 5.8 国际订单状态
- ▲ 6 物理架构
  - 6.1 IDC机房
  - 6.2 国内网站
  - 6.3 国内数据库
  - 6.4 国际网站
  - 6.5 国际数据库
  - 6.6 集群清单
  - 6.7 域名清单
- ▷ 7 接口架构

- 主营业务
- 商务主体：供、采、平台
- 商务模式：4%。
- 组织架构：人数、组织
- 商务运作模式：顶层业务运转
- 关键业务流程：  
预订流程、订单流程、供应流程、结算流程、账号流程
- 附档业务资料

## 2 企业商务模型

- 2.1 商务模式
- 2.2 商务主体
- 2.3 主营业务
- 2.4 组织结构
- 2.5 商务运作模型
- ▲ 2.6 国内机票业务流程
  - 2.6.1 预订流程
  - 2.6.2 订单处理流程
  - 2.6.3 产品供应流程
  - 2.6.4 财务结算流程
  - 2.6.5 账户管理流程
- ▲ 2.7 国际机票业务流程
  - 2.7.1 预订流程
  - 2.7.2 订单处理流程
  - 2.7.3 产品供应流程
  - 2.7.4 财务结算流程
  - 2.7.5 账户管理流程
- 2.8 附档资料





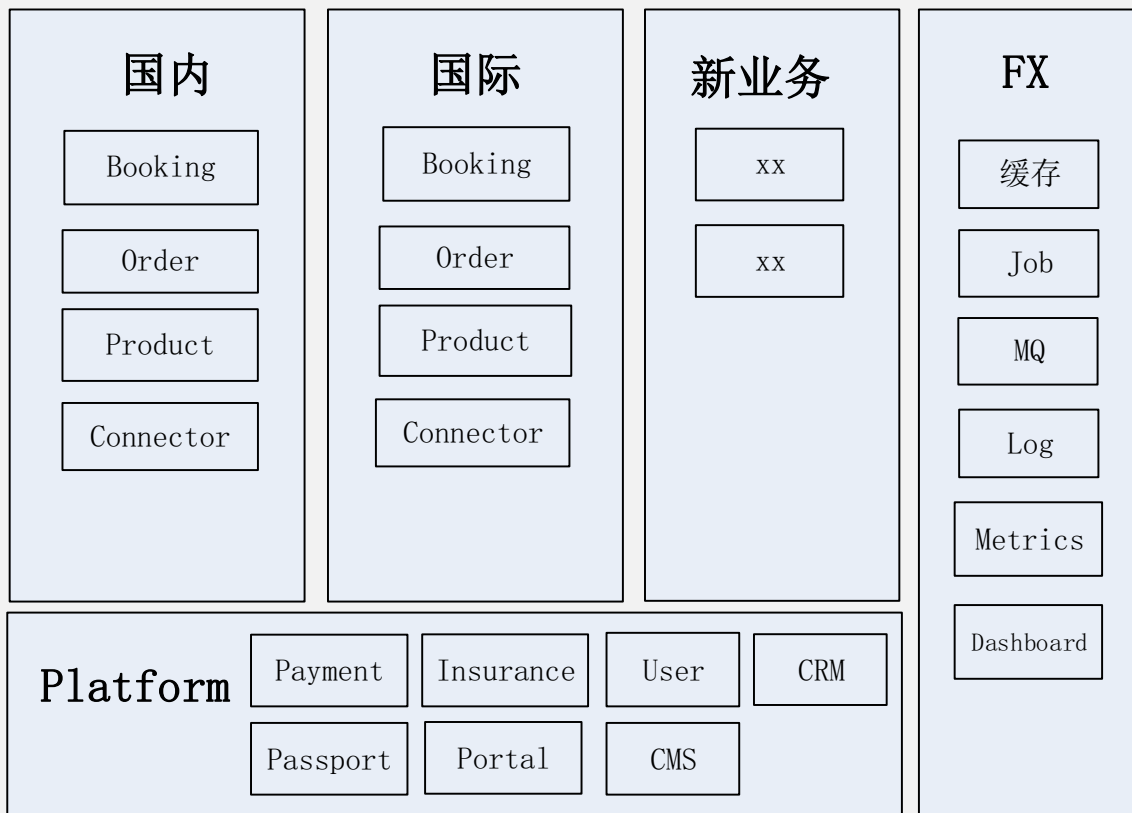
- 微服务设计与康威定律
- 组织架构=》系统架构
- 业务架构=》组织架构

- 功能
- 角色
- 权限
- 问题：先有功能、先有角色？

### 3.1 国内全部功能

采购商的功能：

模块	功能	备注
系统管理	资料信息	
	投诉, 建议	
机场服务	代换登机牌供应管理	
	已确认订单代换处理	
	换登机牌管理	
	代换登机牌	
采购机票管理	PNR 导入创建订单	
	航班查询及预订	
	团队票申请	
	申请改签及升舱查询	
在线订单管理	三字代码查询	
	当日最新订单	
	所有订单查询	
	采购报表下载	
	自由转账	
	退票相关查询	
	申请行程单及查询	
	行程单领用及管理	
	保险管理	
短信平台	短信充值	
	短信发送	
	短信发送历史	
常旅客管理	常旅客添加	
	常旅客查询/修改	
常用软件下载	机票软件下载	

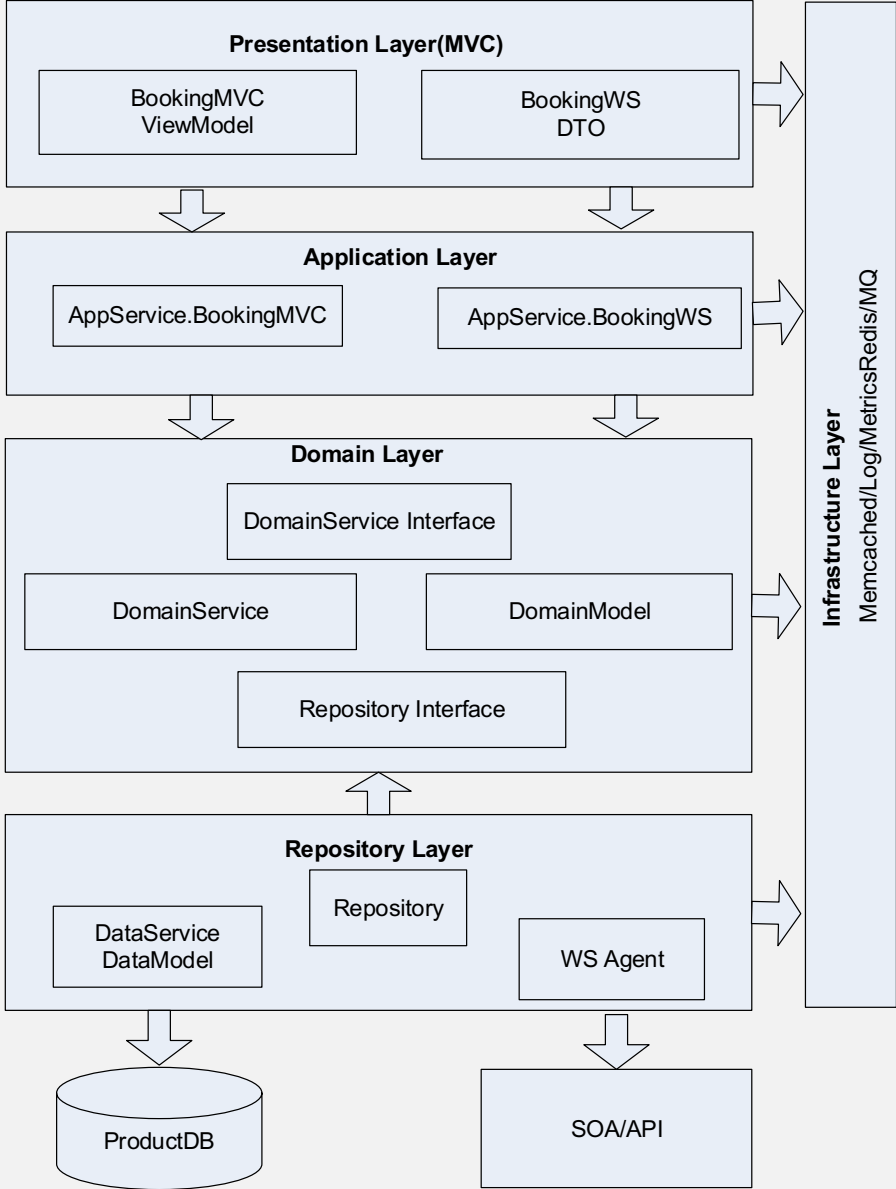
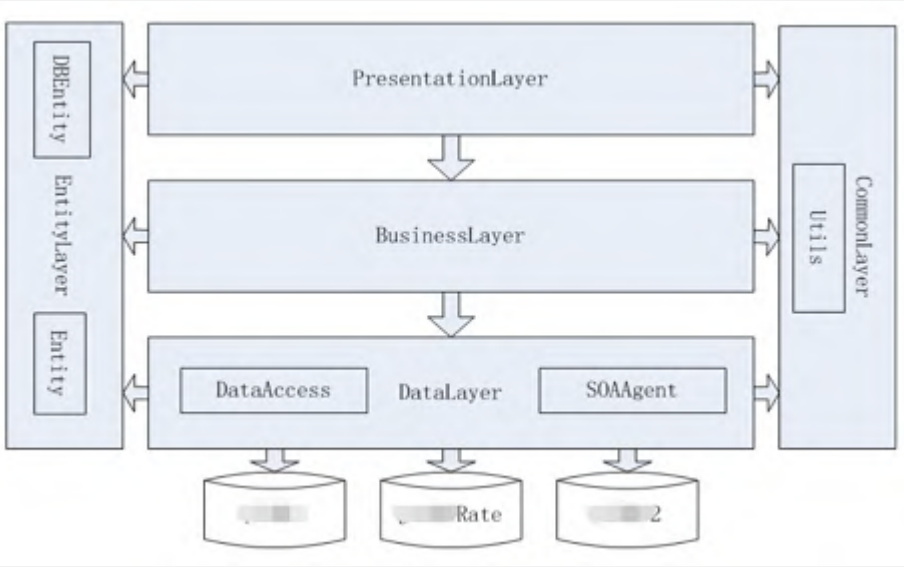


- 产品线=》子系统=》应用
- 应用编号：102010
- 应用名称：  
Flight.Booking.xxx

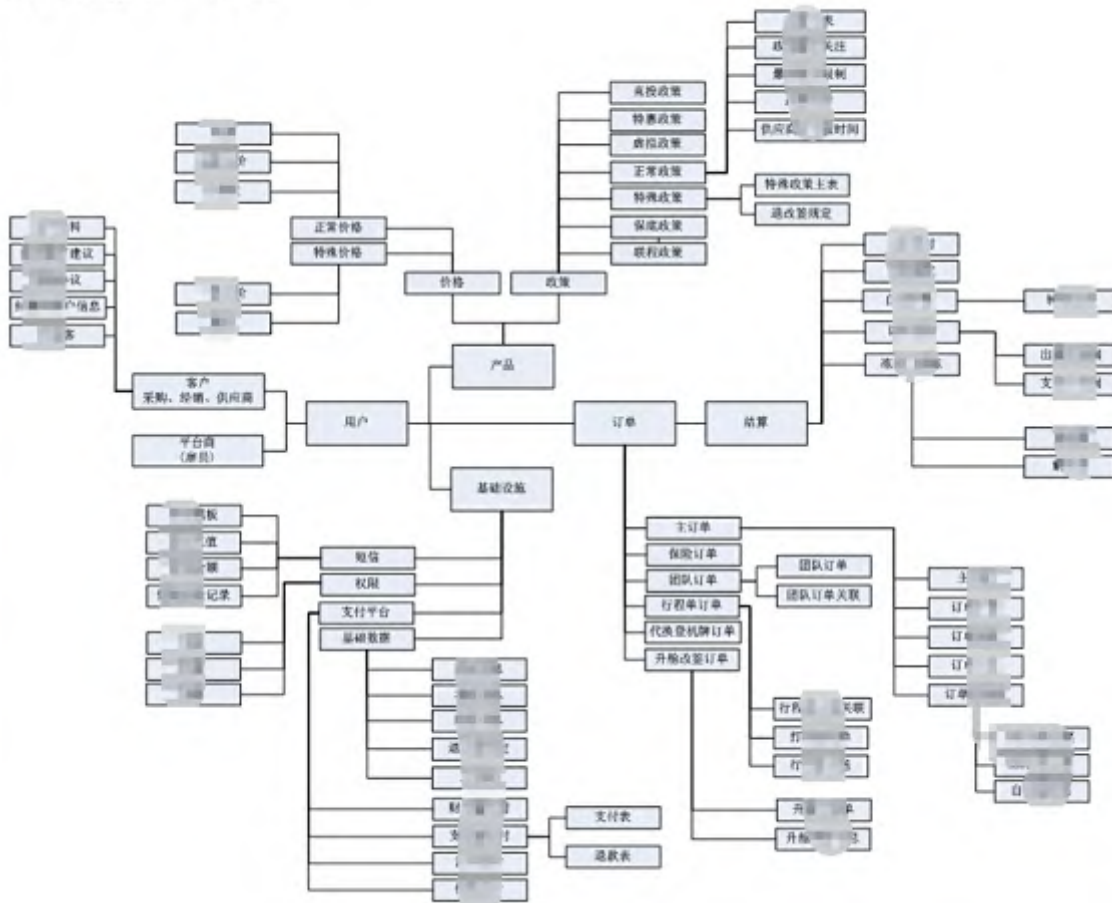
**微服务在哪儿？**

- 领域架构
- 三层架构：IPO

### 微服务的调用应该放哪一层？



### 5.3 国内 E-R 图



- 100多个库, 1万多张表
- 可大可小的设计
- 企业数据=》数据设计=》数据库设计
- 一库一服务是趋势, 不是当前最佳实践

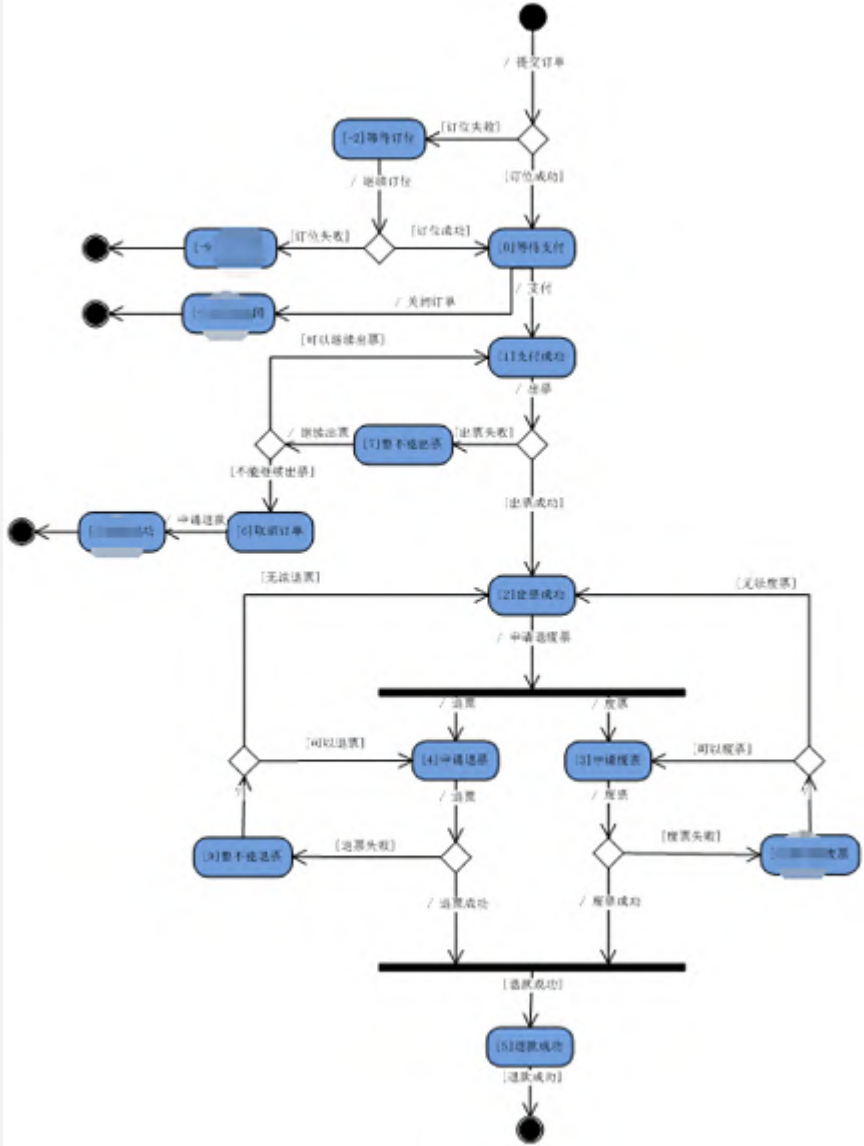
## 微服务设计哪儿?

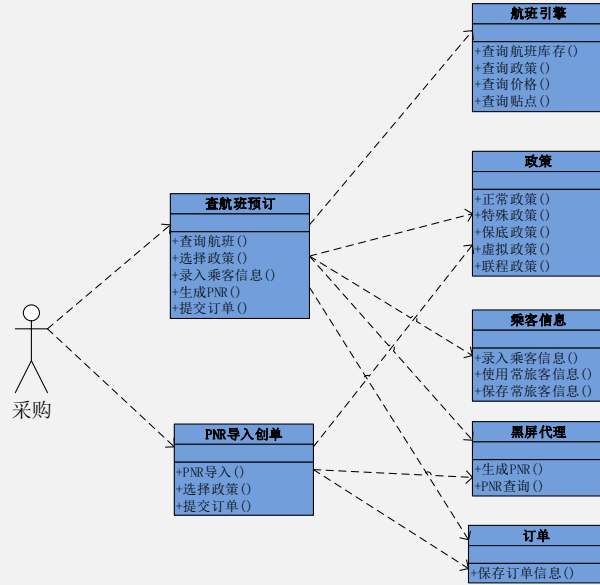
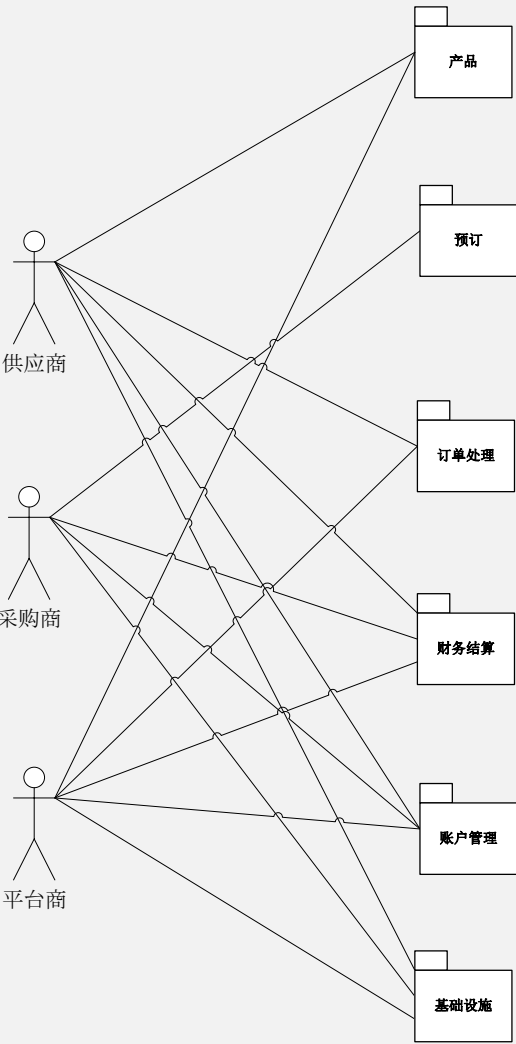
## 9.4 数据库规划

数据库	说明	备注
FitProductDB	国内机票产品库	包括政策和基础价格
FitOrderDB	国内机票订单库	
FitCommonDB	国内机票公共库	
FitAccountDB	国内机票结算库	
FitProductLogDB	国内机票产品日志库	
FitOrderLogDB	国内机票订单日志库	
FitAccountLogDB	国内机票结算日志库	
IFitProductDB	国际机票产品库	包括政策和基础价格
IFitOrderDB	国际机票订单库	
IFitCommonDB	国际机票公共库	
IFitAccountDB	国际机票结算库	
IFitProductLogDB	国际机票产品日志库	
IFitOrderLogDB	国际机票订单日志库	
IFitAccountLogDB	国际机票接口日志库	
UserDB	用户库	将来合成为一个库，现在有国内、国际两个库。
EmpDB	雇员库	同上
MDM	基础主数据	同上
CMS	网站内容管理库	同上
CRM	客户关系管理库	同上
PaymentDB	支付库	同上
PaymentLogDB	支付日志库	同上
InsuranceDB	保险库	同上
ExDB	框架库	

- 状态图是数据与行为的互动
- 关键数据变迁与核心业务流程
- 变迁的过程与微服务

# 微服务在哪儿？

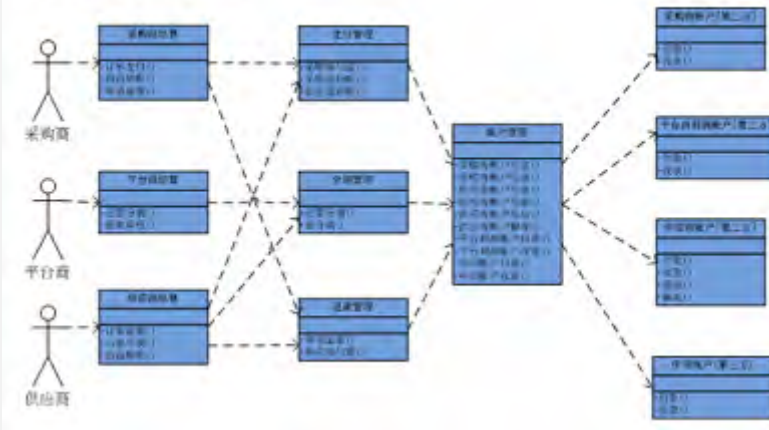




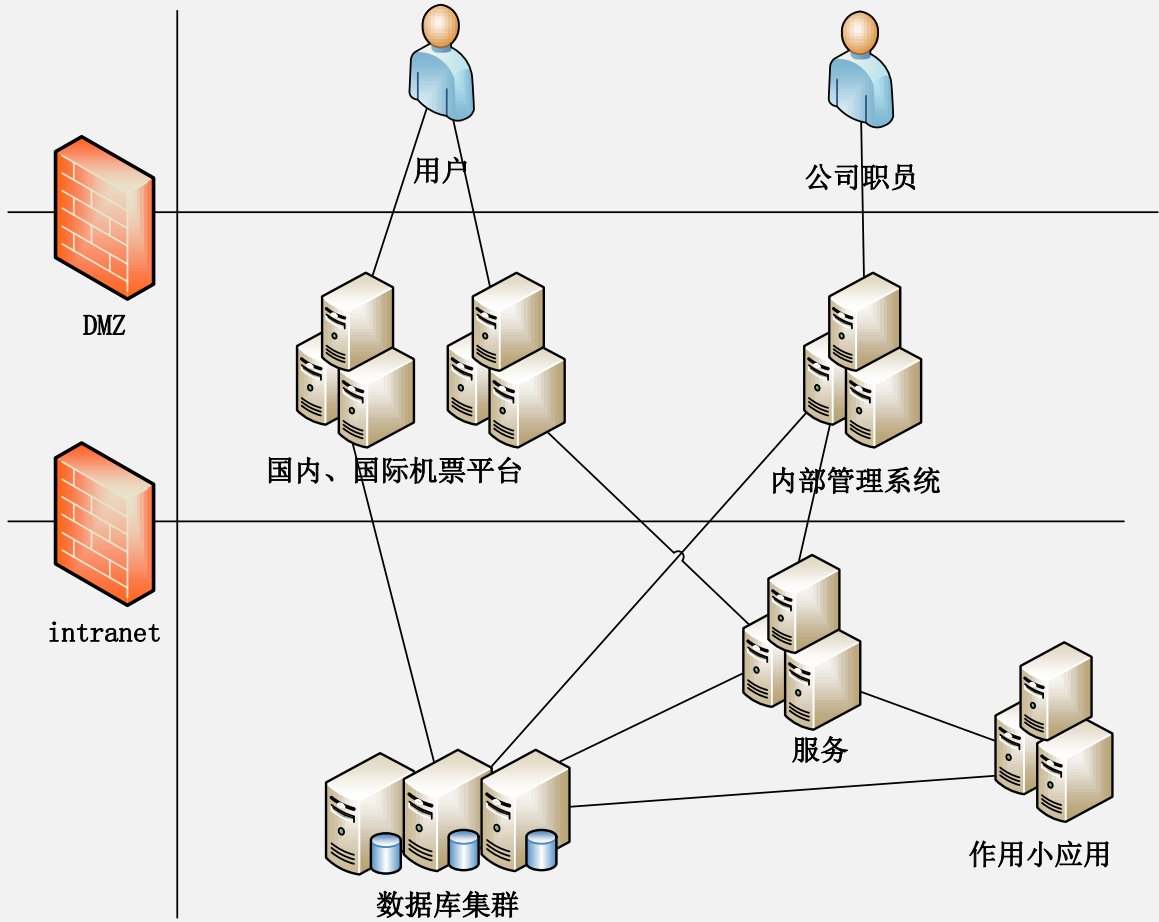
- 业务流程与领域模型
- 业务活动与程序方法

## 微服务在哪儿？

### 8.2.3 财务结算模型

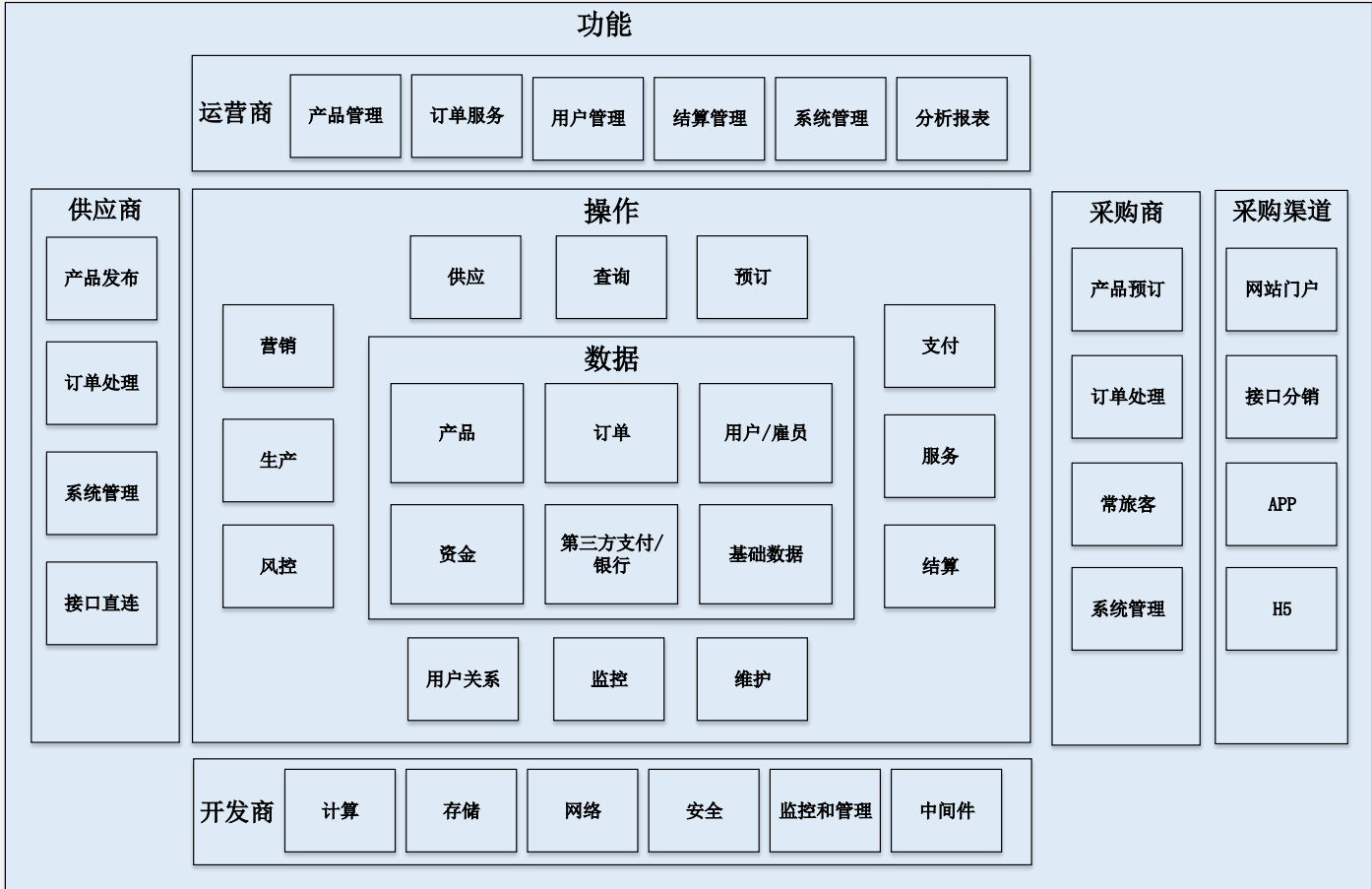






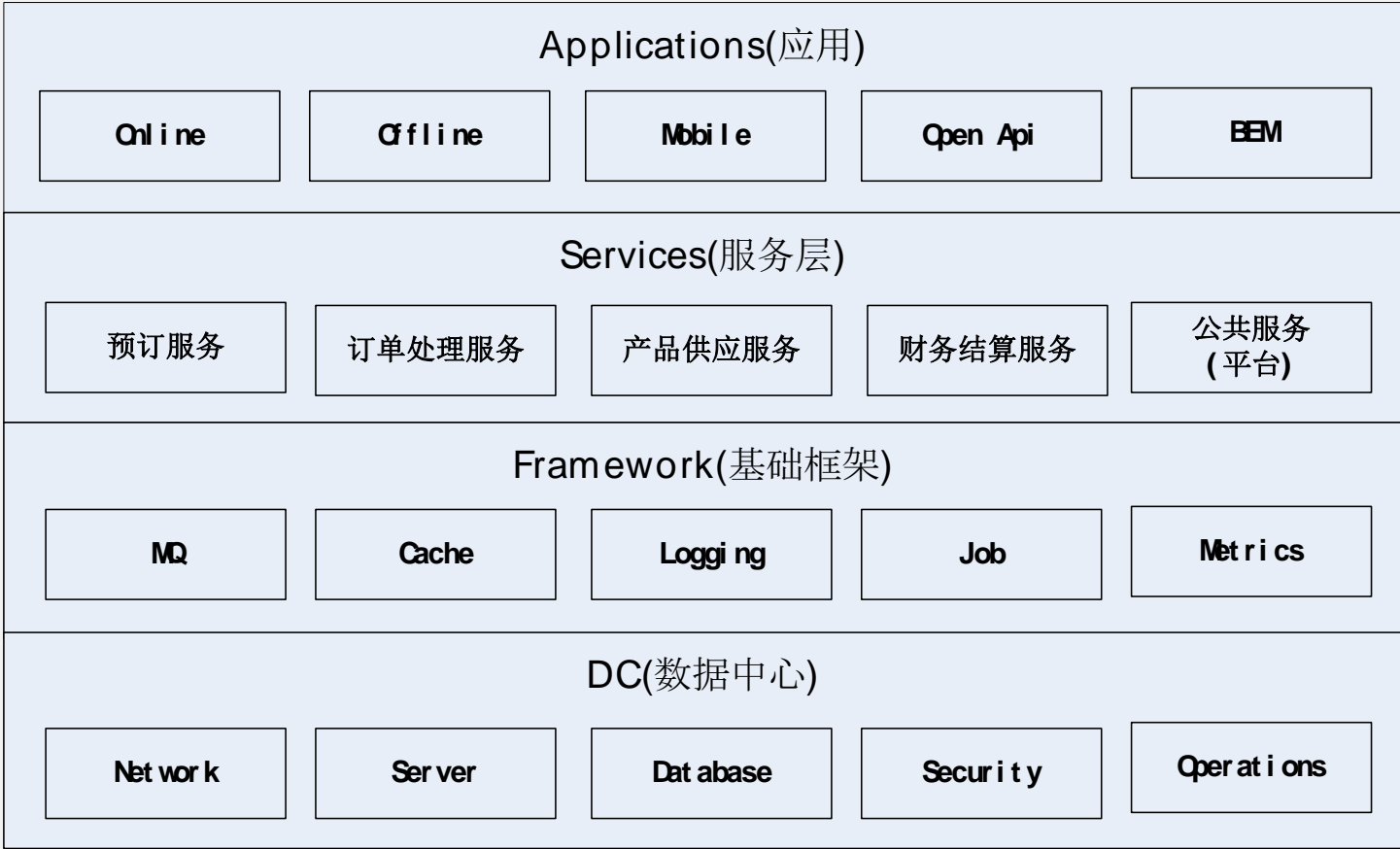
- IDC机房
- 网站物理架构
- 数据库物理架构
- 集群架构
- 域名架构

**微服务部署在哪儿？**



功能=》用户界面  
操作=》服务  
数据=》数据存储

## 微服务在哪儿？



服务的归类  
下层为上层服务  
以客户为中心

微服务在哪儿？

# 微服务与单个项目架构

- 功能需求
- 用例图、用例活动图
- 领域图
- 接口设计
- 分层设计
- 数据库设计
- 物理设计

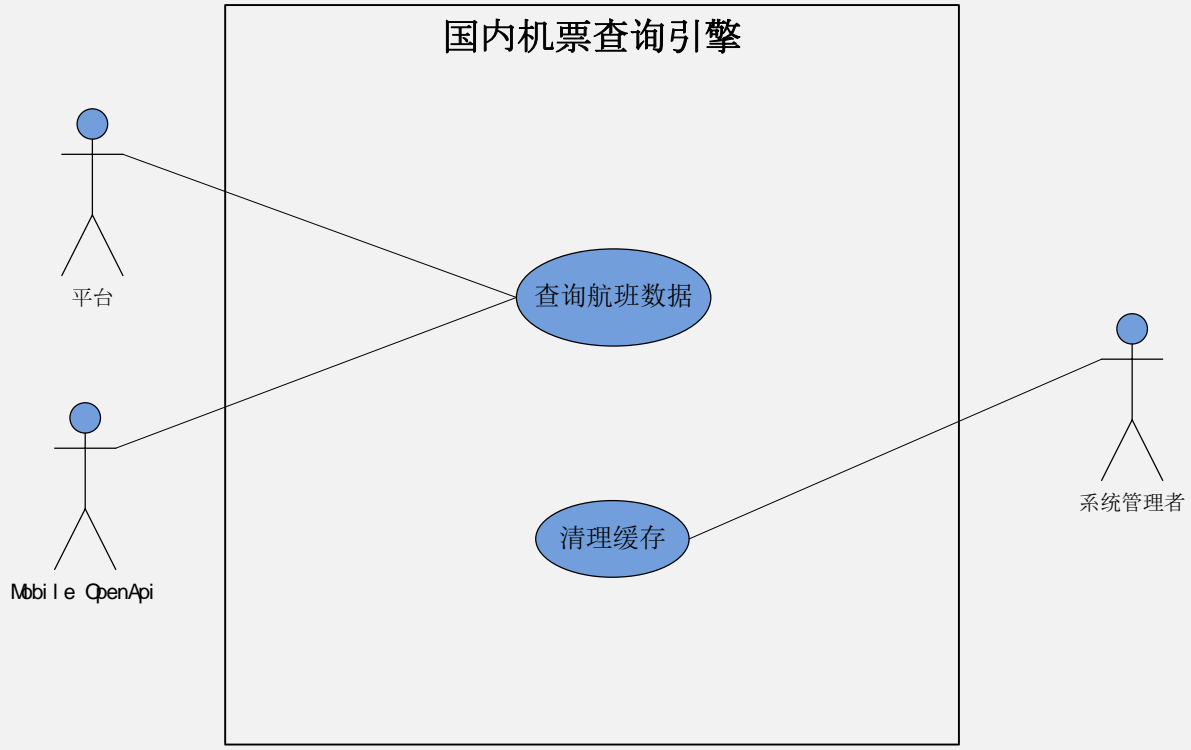
- ▷ 1 系统概述
- 2 设计约定
- ◀ 2.1 关键需求
  - ▷ 2.1.1 功能需求
  - 2.1.2 非功能需求
- 2.2 约束
- ◀ 3 概要设计
  - ◀ 3.1 设计目标和思路
    - 3.1.1 设计目标
    - 3.1.2 设计思路
  - ◀ 3.2 功能设计
    - 3.2.1 用例视图
    - ▷ 3.2.2 用例说明和用例场景
  - 3.3 外部依赖视图
  - 3.4 逻辑架构
  - 3.5 领域设计
  - ▷ 3.6 接口设计
  - 3.7 数据库设计
  - ◀ 3.8 物理架构
    - 3.8.1 应用集群
    - 3.8.2 数据库服务器
    - 3.8.3 App设计
    - 3.8.4 域名
    - 3.8.5 应用部署图
    - 3.8.6 Memcached服务器
- ▷ 4 非功能性设计

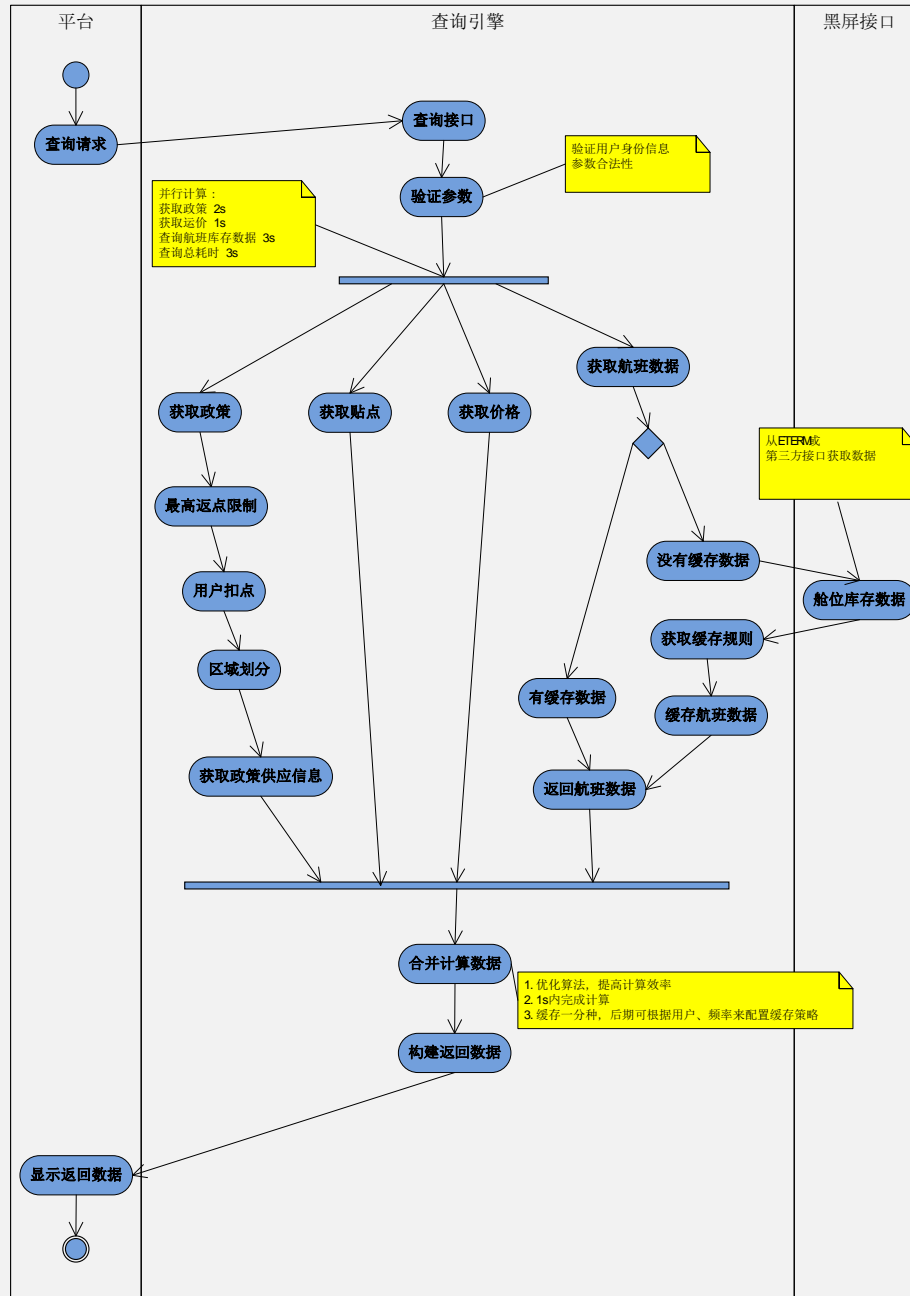
## 2.1.1 功能需求

### 2.1.1.1 功能清单

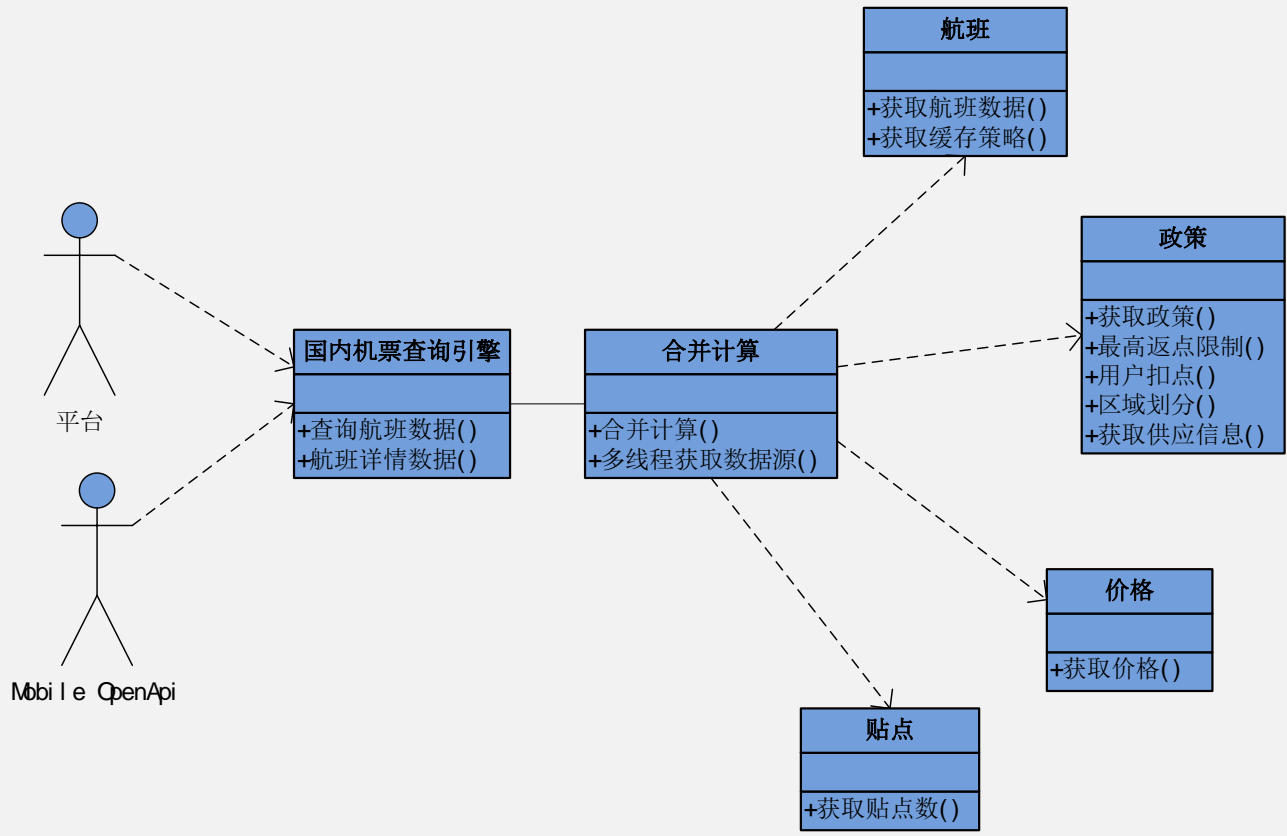
功能模块	主要功能	功能描述
政策获取	航班查询（触发）	
	航班查询	从黑屏及航班接口中查出符合条件航班
	取政策	从政策库取出政策
	取供应信息	出票标识，退票标识，出，退时间
	计算舱位价格	获取不同舱位的价格
	政策限制规则	地域限制、最高返点限制
	显示查询结果	
缓存策略管理	城市级别管理	城市级别，提供配置调整功能
	缓存管理	不同城市级别，缓存时间调整管理
APP 采购贴点	贴点	











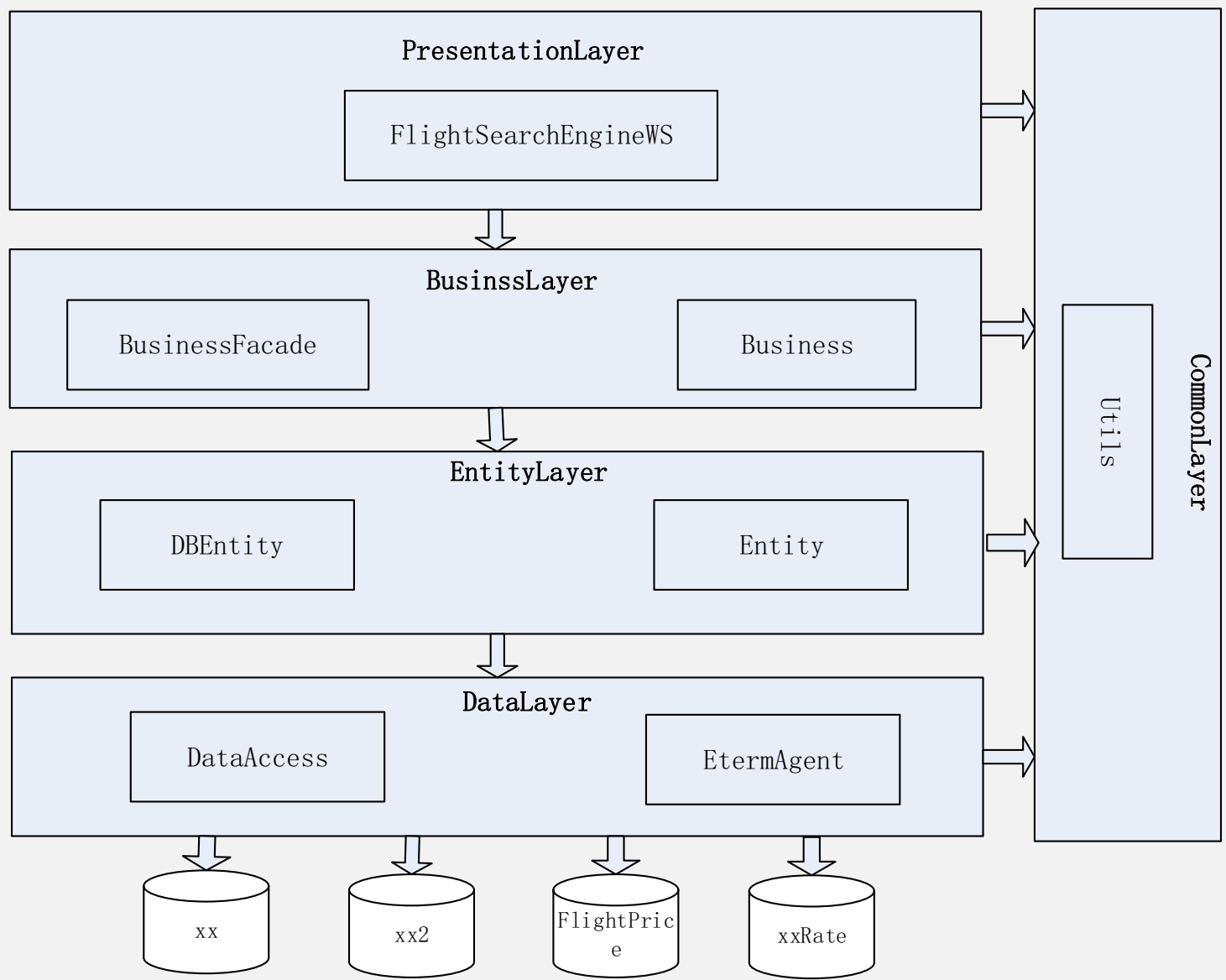
- 交互、连接、契约
- 实施目标
- Request/Response模式
- IPO关注输入输出

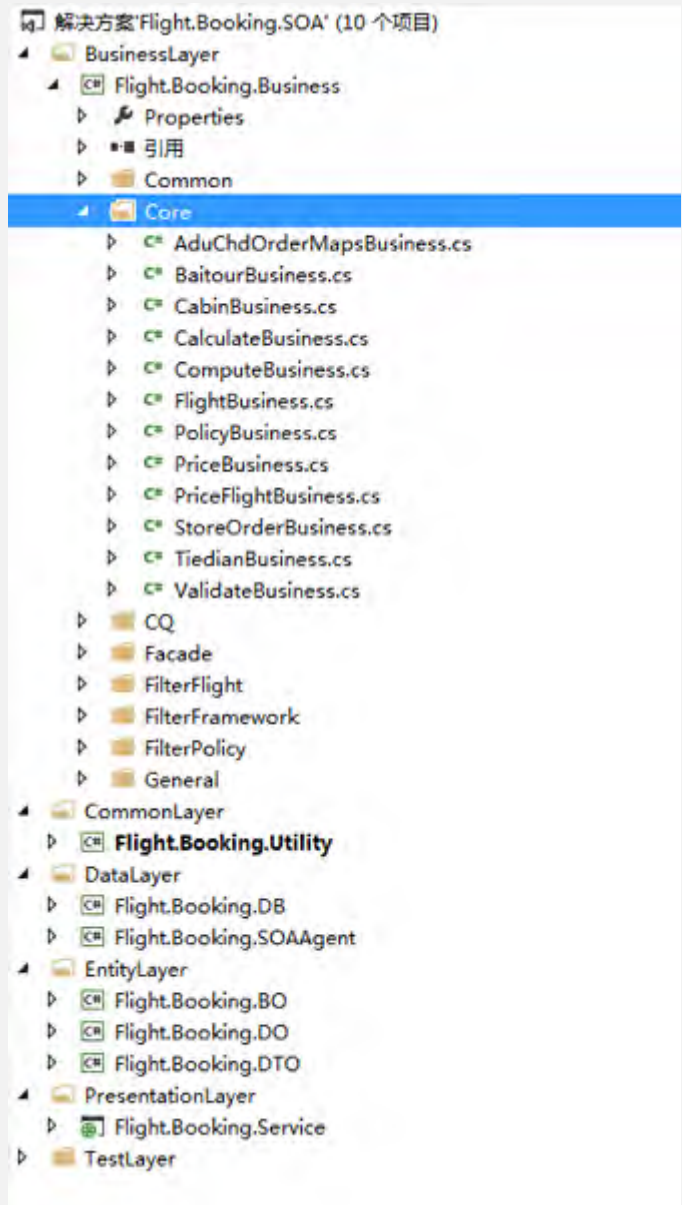
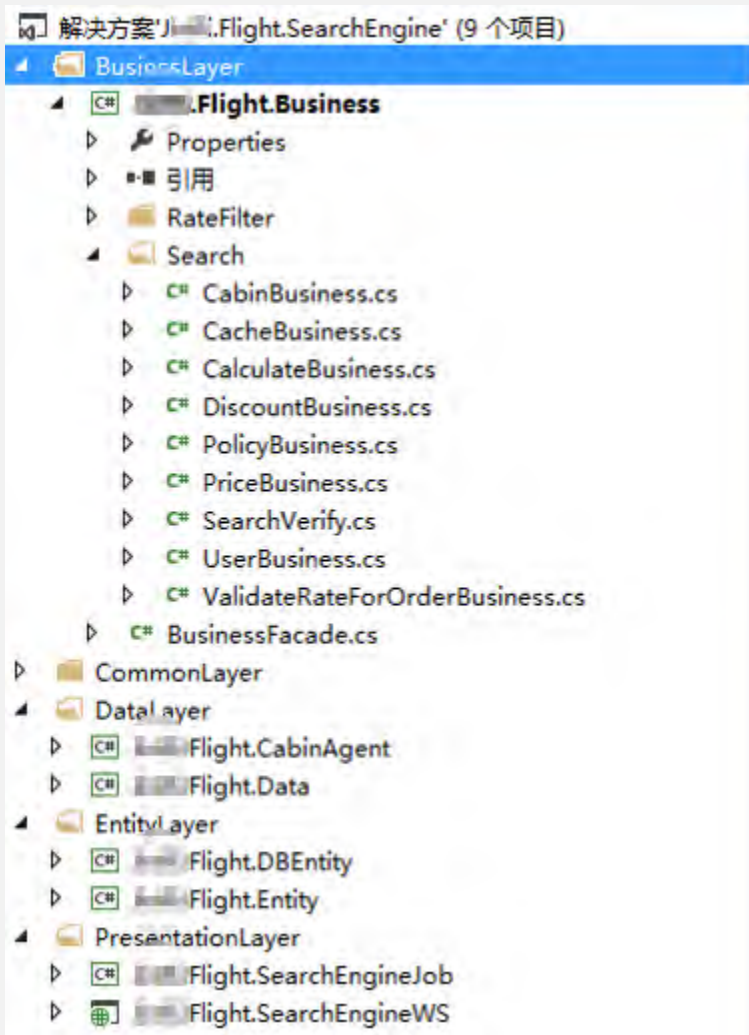
### 3.6.1 查询航班数据接口

- ◆ 接口名称: 航班查询数据接口
- ◆ 接口描述: 接收输入方的请求参数, 合并计算并返回航班查询数据
- ◆ 接口提供者信息: 国内机票数据引擎
- ◆ 接口使用者信息: 平台: mobileOpenApi
- ◆ 接口方式: WCF
- ◆ 接口周期: 实时

请求消息对象					
节点名	英文名称	中文名称	数据类型	必填	备注
FlightSearchRequest	UserName	用户名	String	T	用户名
	RequestType	请求类型	String	T	请求类型 (OpenApi, Web)
	Scity	出发城市	String	T	出发城市三字码(CSX)
	Ecity	到达城市	String	T	到达城市三字码(CSX)
	RateFlag	取票标记	String	F	获取取票 True 不获取取票 False(暂不提供)
	SDate	出发日期	String	T	出发日期 yyyy-MM-dd
	AirLine	航司二字码	String	F	航司二字码(CA)
	FlightNo	航班号	String	F	航班号(MU1234)
	PolicyNum	舱位返回取票条数	Int	F	返回取票条数, 默认 1 个
	Cabin	舱位类型	String	F	舱位类型 A: 所有舱位(默认) B: 经济舱 C: 头等舱 D: 豪华舱 E: 公务舱

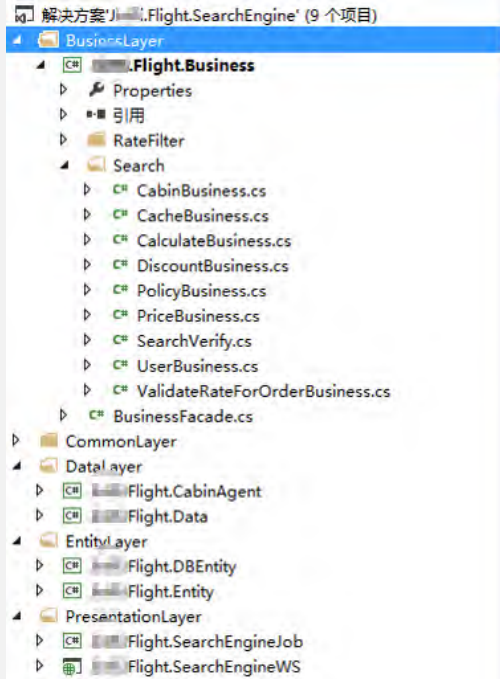
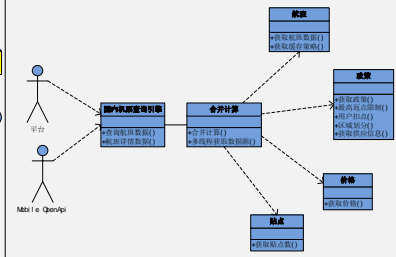
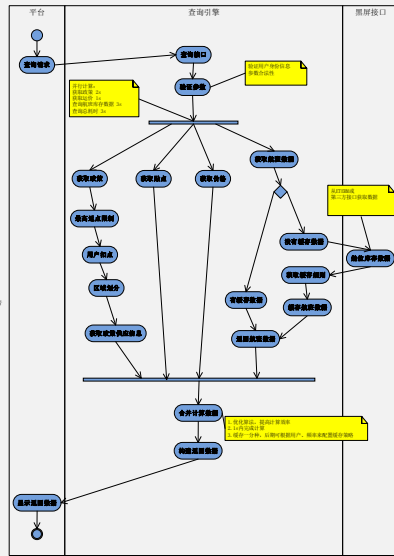
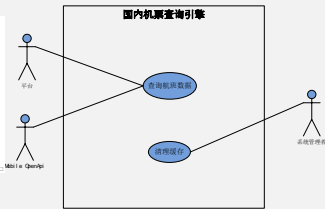
响应消息对象					
节点名	英文名称	中文名称	数据类型	必填	备注
FlightSearchResponse	Success	成功	bool	T	成功标志
	ErrMsg	错误信息	String	F	错误描述
Flight	Flights	航班集合	List<Flight>	T	
	SCity	出发城市	String	T	PEK
	Ecity	到达城市	String	T	CSX
	SDate	出发日期	String	T	yyyy-MM-dd
	AirLine	航司二字码	String	T	MF





2.1.1 功能需求

功能名称	功能描述
航班查询	根据条件查询航班
航班预订	从航班搜索结果中选中符合条件的航班
取消预订	从航班搜索结果中取消预订
航班变更	变更航班、变更舱位、出、退行程
计算航班价格	根据航班搜索结果计算价格
计算航班里程	根据航班搜索结果计算里程
保存航班管理	城市列表、提供配置城市
API 文档管理	提供管理



功能需求=》用例图=》活动图=》领域模型=》代码实现

# 探讨几点

- 业务架构：业务流程、业务活动
- 领域模型
- 数据变迁图
- 接口架构
- 程序方法

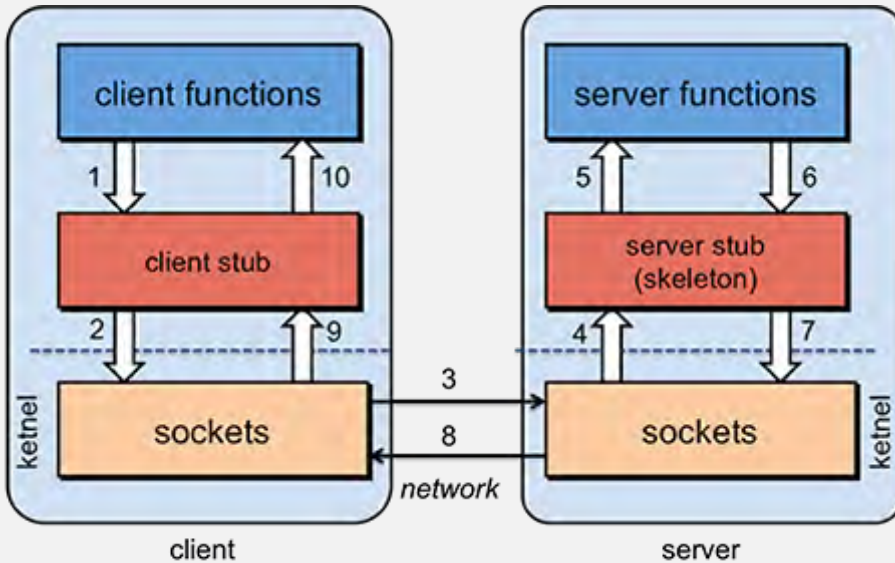




- 拆分原则：业务发展阶段，企业规模，技术复杂度
- 变化规律：微服务不停长大，不停拆小
  - 人：控制复杂度，大服务=》微服务，从大到小
  - 事：复杂度累加，微服务=》大服务，从小到大

# 微服务技术实现

- 微服务=Client/Server+HTTP+数据+JSON
- 通讯=通讯双方+线路+报文+编码
- RPC=Client/Server+通道+数据+编码
- 沟通=沟通双方+邮件/电话+信息+中/英
- 快递=买卖双方+路+货+打包

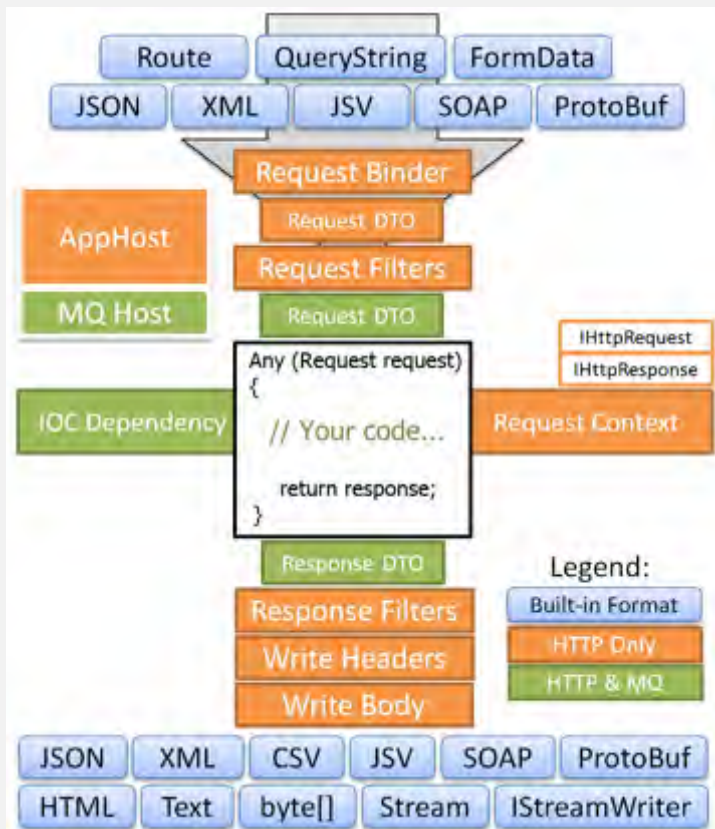


### 微服务优点：

- 轻量级、粒度小、简单、兼容
- RESTful : URI+HTTP Method
- Error Code : HTTP Code
- 多语言支持、数据类型
- 人与机器都可读
- 浏览器Javascript

- 框架需求：
  - 调试
  - 测试
  - 文档
  - JSON元数据
  - 多协议：JSON/XML/Protobuf
  - 更多：性能/框架轻/易学易用/容易改造
- 开源改造：
  - **ServiceStack**/OWIN/Spring Cloud

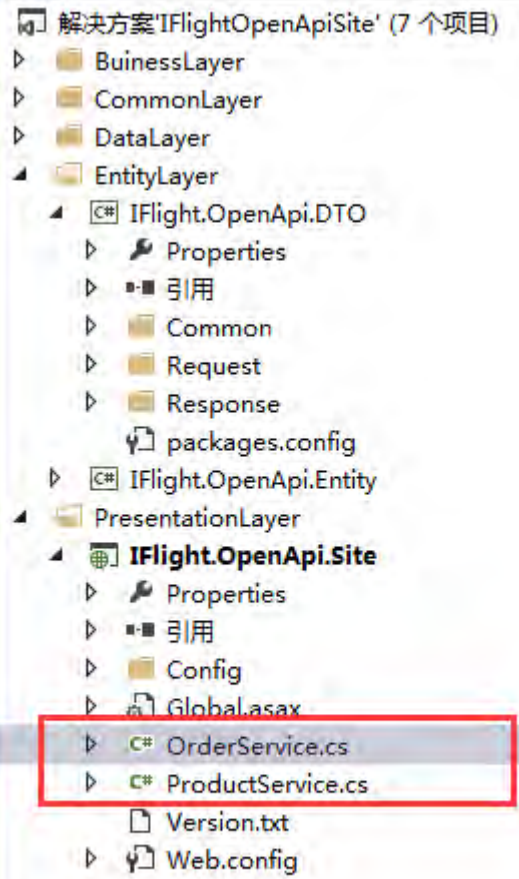
# • 服务端架构



# • 客户端架构



- 业务逻辑代码的开放
- 路由配置
- 开通Protobuf & Swagger



```
internal class ServiceAppHost : AppHostBase
{
    1 1 引用
    public ServiceAppHost ()
        : base("IFlightOpenAPI", typeof(ServiceAppHost).Assembly)
    {
        Routes.Add<SearchFlightPriceRequest>("/SearchService/FlightPrice", "POST", "航班查询");
        Routes.Add<SearchPnrPriceRequest>("/SearchService/PnrPrice", "POST", "PNR 查询");
        Routes.Add<SearchPnrTextPriceRequest>("/SearchService/PnrTextPrice", "POST", "PNR 文本查询");
        Routes.Add<SearchPolicyRequest>("/SearchService/Policy", "POST", "政策查询");
        Routes.Add<SearchAirRulesRequest>("/SearchService/AirRules", "POST", "退改签规则查询");

        Routes.Add<CreateOrderRequest>("/OrderService/Create", "POST", "下单");
        Routes.Add<CreateOrderByPnrRequest>("/OrderService/CreateByPnr", "POST", "PNR 下单");
        Routes.Add<CreateOrderByPnrTextRequest>("/OrderService/CreateByPnrText", "POST", "PNR 文本下单");
        Routes.Add<AdvancePayOrderRequest>("/OrderService/AdvancePay", "POST", "预付支付");
        Routes.Add<AutoPayOrderRequest>("/OrderService/AutoPay", "POST", "代扣支付");
        Routes.Add<RefundOrderRequest>("/OrderService/Refund", "POST", "退票申请");

        Plugins.Add(new ProtoBufFormat());

        JsConfig.EmitCamelCaseNames = false;
        Plugins.Add(new SwaggerFeature());
    }
}
```

- 多协议
- 文档
- 元数据
- 调试工具

## IFlightOpenAPI

The following operations are supported. For a formal definition, please review the Service XSD.

### Operations

- AdvancePayOrderRequest
- AutoPayOrderRequest
- CreateOrderByPnrTextRequest
- CreateOrderRequest
- SearchAirRulesRequest
- SearchFlightPriceRequest
- SearchPnrTextPriceRequest
- SearchPolicyRequest

XML	JSON	JSV	CSV	X-PROTOBUF	SOAP 1.1	SOAP 1.2
XML	JSON	JSV	CSV	X-PROTOBUF	SOAP 1.1	SOAP 1.2
XML	JSON	JSV	CSV	X-PROTOBUF	SOAP 1.1	SOAP 1.2
XML	JSON	JSV	CSV	X-PROTOBUF	SOAP 1.1	SOAP 1.2
XML	JSON	JSV	CSV	X-PROTOBUF	SOAP 1.1	SOAP 1.2
XML	JSON	JSV	CSV	X-PROTOBUF	SOAP 1.1	SOAP 1.2
XML	JSON	JSV	CSV	X-PROTOBUF	SOAP 1.1	SOAP 1.2
XML	JSON	JSV	CSV	X-PROTOBUF	SOAP 1.1	SOAP 1.2
XML	JSON	JSV	CSV	X-PROTOBUF	SOAP 1.1	SOAP 1.2

### Clients Overview

#### XSDS:

- Service Types
- Wcf Data Types
- Wcf Collection Types

#### WSDLs:

- soap11
- soap12

#### Plugin Links:

- Swagger UI

Parameter	Value	Description	Parameter Type	Data Type
Id	<input type="text"/>	订单ID号	path	string
CustomerName	<input type="text"/>	客户名	form	string
IsTakeaway	<input type="text"/>	是否已取餐	form	string
CreateDate	<input type="text"/>	创建订单日期	form	string
StatusCode	<input type="text"/>	订单状态	form	string
OrderItemList	<input type="text"/>	订购的产品列表	form	string
body	<input type="text"/>		body	Model Model Schema

```

{
  "Id": "123",
  "CustomerName": "",
  "IsTakeaway": false,

```

- **Swagger :**
  - JSON元数据
  - 可调试
  - 文档
- **传统调试 :**
  - UT
  - 调用者

The screenshot shows the Swagger UI interface for 'IFlight OpenAPI'. At the top, there's a header with 'Swagger UI', a URL 'http://localhost:5399/resources', an 'api\_key' field, and an 'Explore' button. Below the header, the API name 'IFlight OpenAPI' is displayed, along with navigation options: 'Show/Hide', 'List Operations', 'Expand Operations', and 'Raw'. The main content area shows two API endpoints under the 'OrderService' group:

- POST /OrderService/Create** (下单)
- POST /OrderService/CreateByPnrText** (PNR 文本下单)

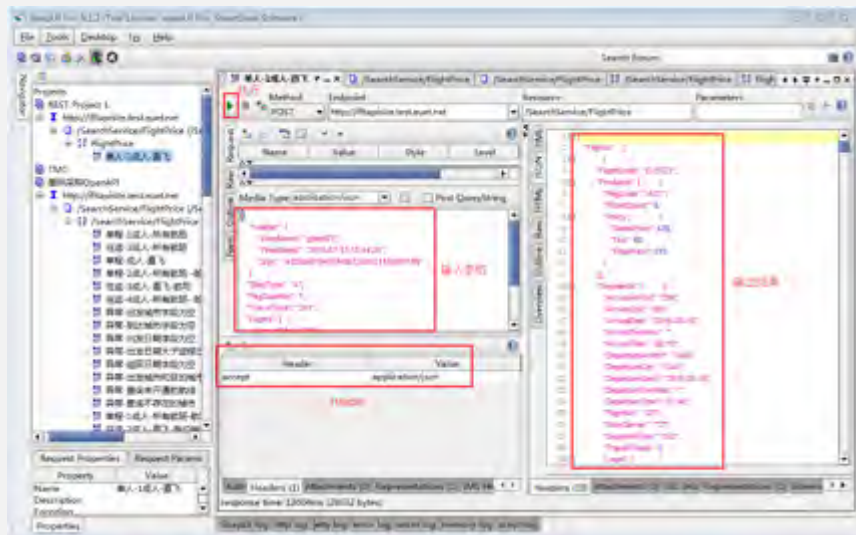
The selected endpoint, 'POST /OrderService/CreateByPnrText', is expanded to show its details. A red box highlights the 'Response Class (Status)' section, which includes a 'Model | Model Schema' link and a code block for 'CreateOrderByPnrTextResponse':`CreateOrderByPnrTextResponse {  
 Code (int): 返回码,  
 Message (string, optional): 返回消息,  
 ContactInfo (ContactInfo, optional): 联系人信息,  
 OrderNo (string, optional): 订单号,  
 OrderRemark (string, optional): 平台备注,  
 OrderState (int): 订单状态,  
 Passengers (Array[OrderPassenger], optional): 旅客列表,  
 PayPrice (double): 订单总金额,  
 PNR (string, optional): PNR,  
 PNRText (string, optional): PNR内容,  
 PolicyID (string, optional): 政策ID,  
 Remark (string, optional): 订单备注,  
 Segments (Array[OrderSegment], optional): 航程列表  
}`

Below the response class, there's a 'Response Content Type' dropdown set to 'application/json'. Underneath, the 'Parameters' section is visible, with a table header: 'Parameter', 'Value', 'Description', 'Parameter Type', and 'Data Type'. A 'body' parameter is listed with a 'body' value and 'body' parameter type. A red box highlights the 'body' parameter's value field, which contains a JSON object:`{  
 "ContactInfo": {  
 "Contact": "",  
 "Tel": ""  
 },  
 "FlightCode": "",  
 "Passengers": [  
 {  
 "Birthday": "",  
 "CardNo": "",  
 "CardType": "",  
 "Country": "",  
 "ExpiryDate": ""  
 }  
 ]  
}`

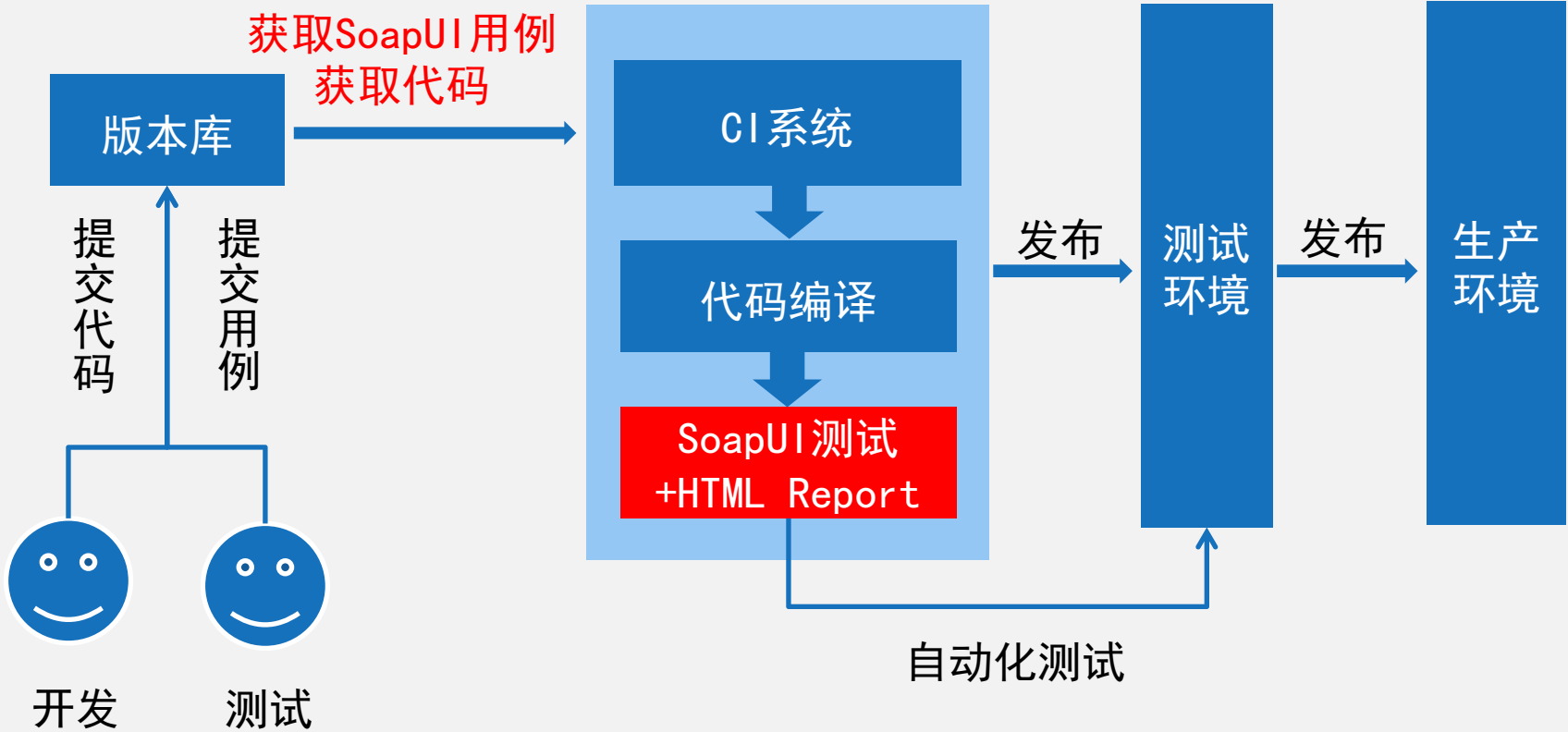
At the bottom of this field is a 'Parameter content type: application/json' dropdown and a 'Try it out!' button. To the right of the 'body' parameter, there's a 'Model | Model Schema' section with a red box around it. It contains a 'Click to set as parameter value' link and a JSON object that matches the one in the 'body' parameter's value field:`{  
 "ContactInfo": {  
 "Contact": "",  
 "Tel": ""  
 },  
 "FlightCode": "",  
 "Passengers": [  
 {  
 "Birthday": "",  
 "CardNo": "",  
 "CardType": "",  
 "Country": "",  
 "ExpiryDate": ""  
 }  
 ]  
}`

A red arrow points from the 'body' parameter's value field to the 'Model | Model Schema' section, indicating the link between the parameter value and the schema definition.





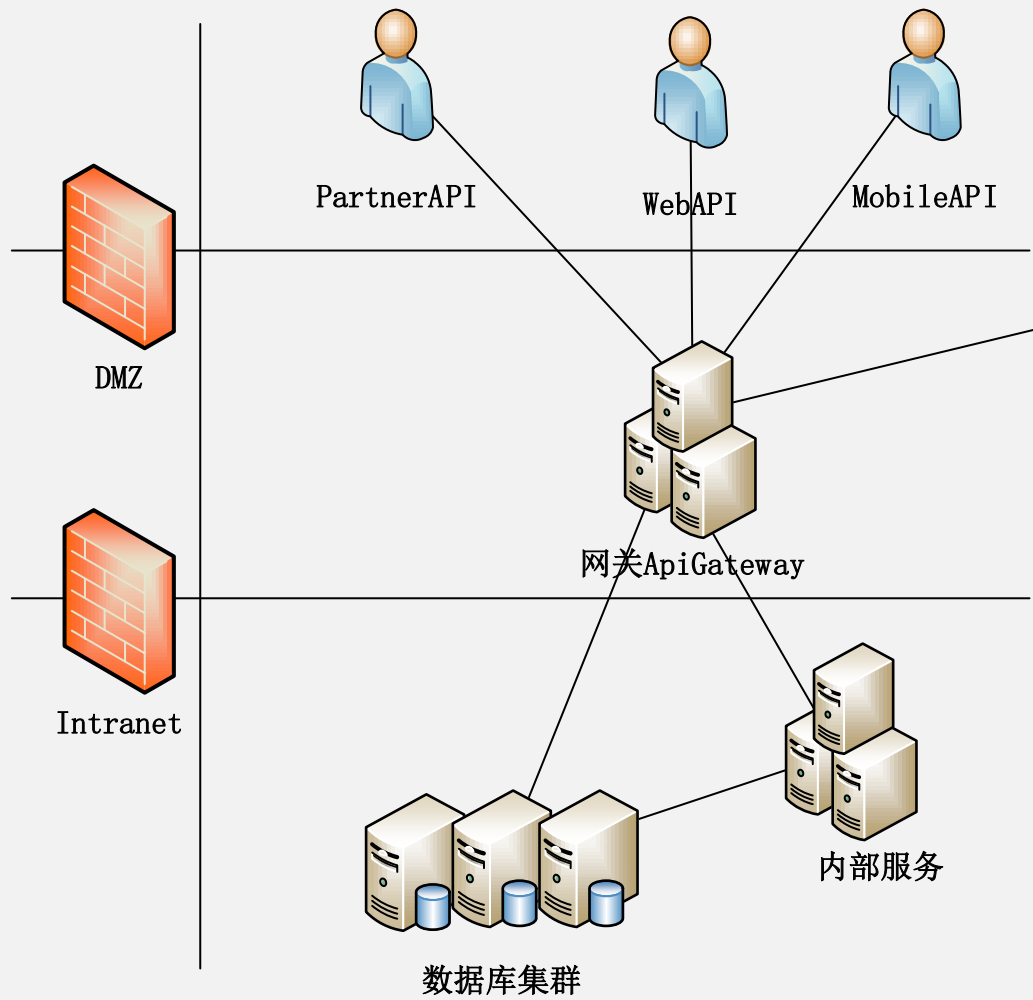
- ✓ 测试工具/测试调用/测试桩/断言
- ✓ 用例管理/测试数据外置/测试报告
- ✓ 一键测试/负载测试/性能测试



- 接口注册与应用
- 接口编号ApiId
- OpenApiName
- 开放选项

The image shows a two-part screenshot of a web application. The left part is a dark-themed sidebar menu with a red header '应用管理'. The menu items are '产品线管理', '子系统管理', '应用管理', and '接口注册', with '接口注册' highlighted. Below it is 'API网关管理'. The right part is a light-themed form titled '编辑接口注册'. The form contains several fields: '接口编号' (11010505), '接口中文名' (下单), '接口路径' (/OrderService/Create), '是否开放' (checked), 'OpenApiName' (CreateOrder), and 'HTTP Method' (POST). There are also text areas for '请求体' and '响应体'. At the bottom are '保存' and '取消' buttons.

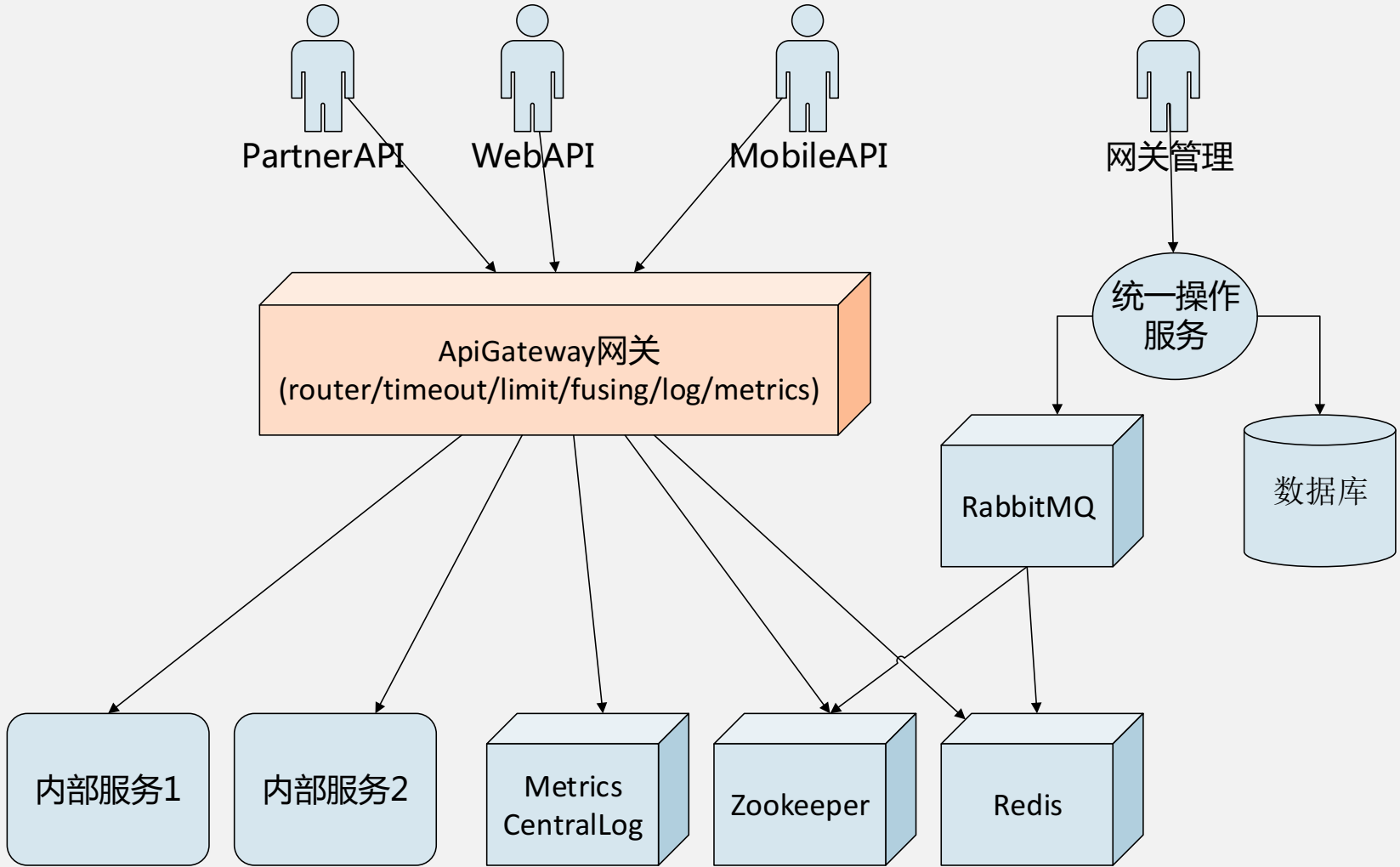
Field	Value
接口编号	11010505
接口中文名	下单
接口路径	/OrderService/Create
是否开放	<input checked="" type="checkbox"/>
OpenApiName	CreateOrder
HTTP Method	POST
请求体	
响应体	



## 为什么要网关？

- 1. 网络隔离
- 2. 用户权限
- 3. 非功能性需求/复用
- 4. 方便测试
- 5. 统一管理

- Route : OpenApiName => 内部服务
- Auth : IP黑名单/角色客户/网关授权
- Timeout : 网关 => 内部服务
- Fusing :
  - 异常阈值 => 断开链路 => 冷却时间 => 链路闭合
  - 保险丝, 无谓消耗
- Limit : 调用频率 > 阈值 => 断开链路
- Metrics&Log : 实时监控和集中式日志



- 网关策略：Timeout/Limit/Fusing
- 绑定API：一对多
- 设置客户例外

The screenshot displays the 'API Gateway Management' interface. On the left, a sidebar menu is visible with 'API Gateway Management' selected, and a sub-menu where 'Gateway Strategy' is highlighted. The main content area is titled 'Gateway Strategy' and contains a table of configurations. A red box highlights the table's data columns.

<input type="checkbox"/>	策略名称	通讯协议	超时时间(秒)	调用频率(次/分)	异常频率(次/分)	熔断冷却(分)	熔断次数	状态	操作
<input type="checkbox"/>	国际机票 OpenAPI	JSON	5	60 / 1	5 / 1	10	3	启用	<a href="#">编辑</a> <a href="#">绑定API</a> <a href="#">设置客户例外</a>
<input type="checkbox"/>	订单服务策略	JSON	30	100 / 1	20 / 1	5	3	启用	<a href="#">编辑</a> <a href="#">绑定API</a> <a href="#">设置客户例外</a>

- 微服务与HttpJob
- 微服务与集中式日志
- 微服务与容器
- 微服务与监控

编辑任务信息 点击关闭 或 按Esc键

*任务名称 :	CancelOrder
组名 :	汇
请求地址 :	http://[redacted]/OrderService/CancelOrder
请求类型 :	POST
JSON请求体 :	<div style="border: 1px solid #ccc; height: 100px;"></div>
开始时间 :	2016-12-30 20:07:56
Trigger类型 :	SimpleTrigger
重复次数 :	-1
间隔时间(c) :	60



- 微服务的调用放到哪一层？
- 微服务从哪里来？
- 微服务架构在哪个业务阶段最佳？
- 微服务与业务架构的关系？

单个项目架构：功能需求=》用例图=》活动图=》领域模型=》代码实现

企业总体架构：企业服务=》业务架构=》领域模型=》面向服务架构=》微服务架构

**THANK YOU**

