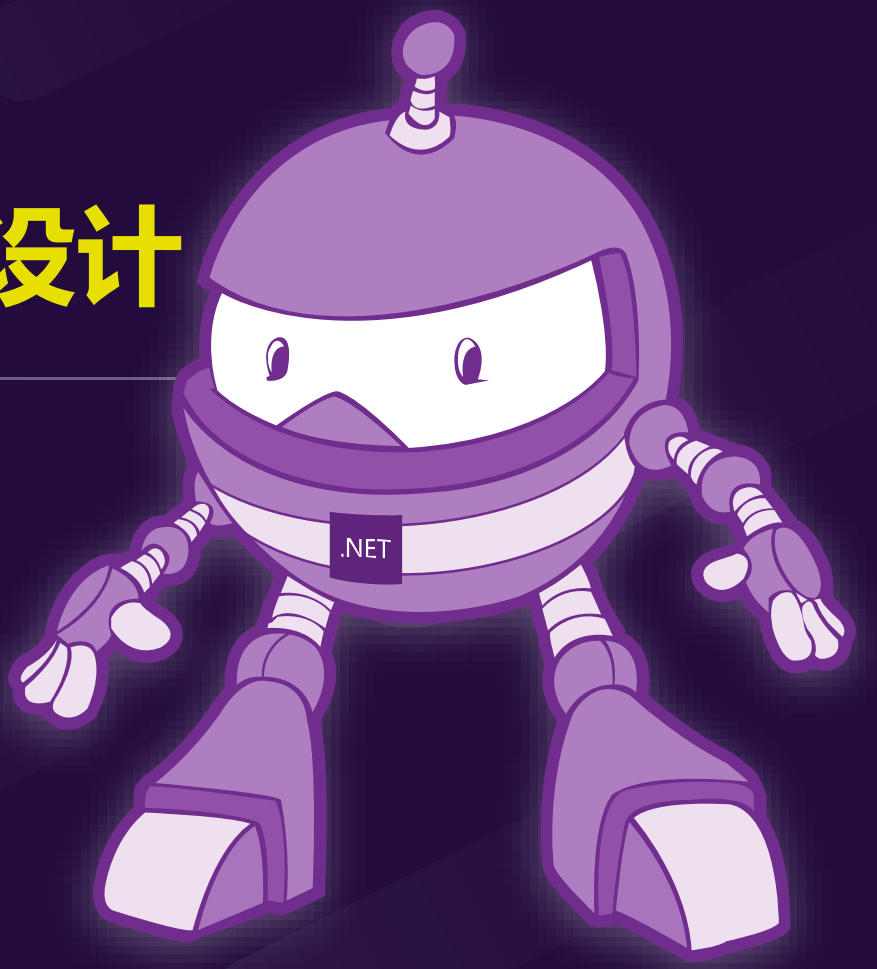
2019中国.NET 开发者峰会
China .Net Conf 2019

.NET Core 2019
微服务快速开发框架的设计

演讲人：朱宗海



$whoami

- Junior system architect @NIO
- Developer background
- Programming for 19 years

- Beginners of microservices and cloud computing (since 2016)

开发框架--what?

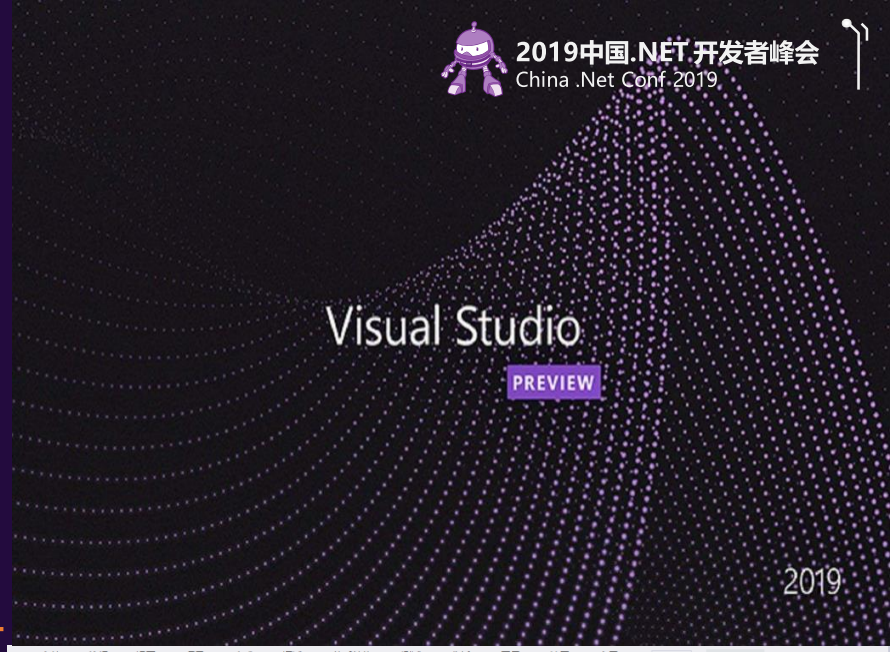
框架（Framework）——整个或部分系统的可重用设计，表现为一组抽象构件及构件实例件间交互的方法；另一种定义认为，框架是可被应用开发者定制的应用骨架。(来自度娘)

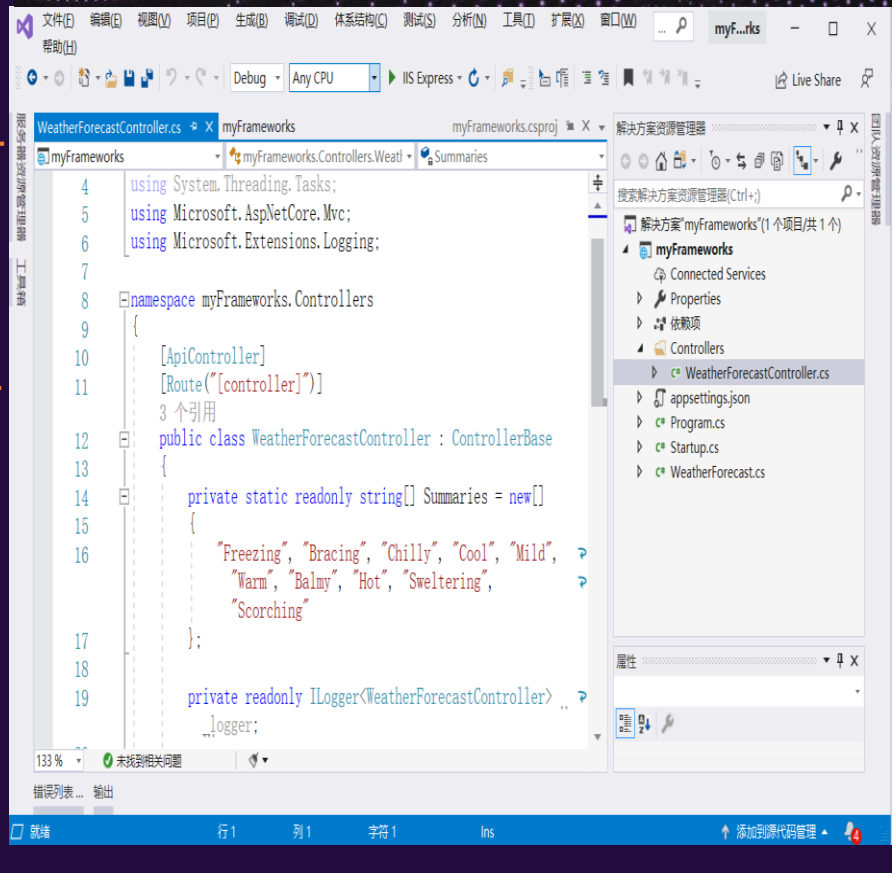
框架的主要特点和要求

- 代码模板化---统一的代码风格
- 重用
- 高内聚（封装）
- 规范
- 可扩展
- 可维护
- 协作开发
- 通用性

分分钟“搞定”一个开发框架

打开电脑，找到“宇宙第一”IDE—Visusal studio的图标，双击启动它，然后请单击“文件”->“新建”->项目，选择.net core web项目……然后，起个名字，选择保存的目录，再“下一步”，再“下一步”……





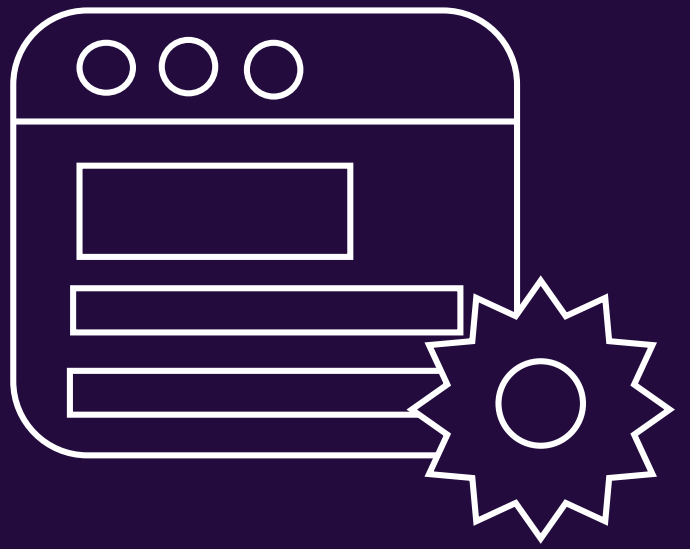
开发框架的构成组件

一个真正可以使用的框架，必须是功能完备的

- 操作数据库
- 记日志
- 缓存的存取
- 常用小工具集
- NoSql的支持
- 对依赖的第三方系统的解耦
- ……（暂时还没有想到）

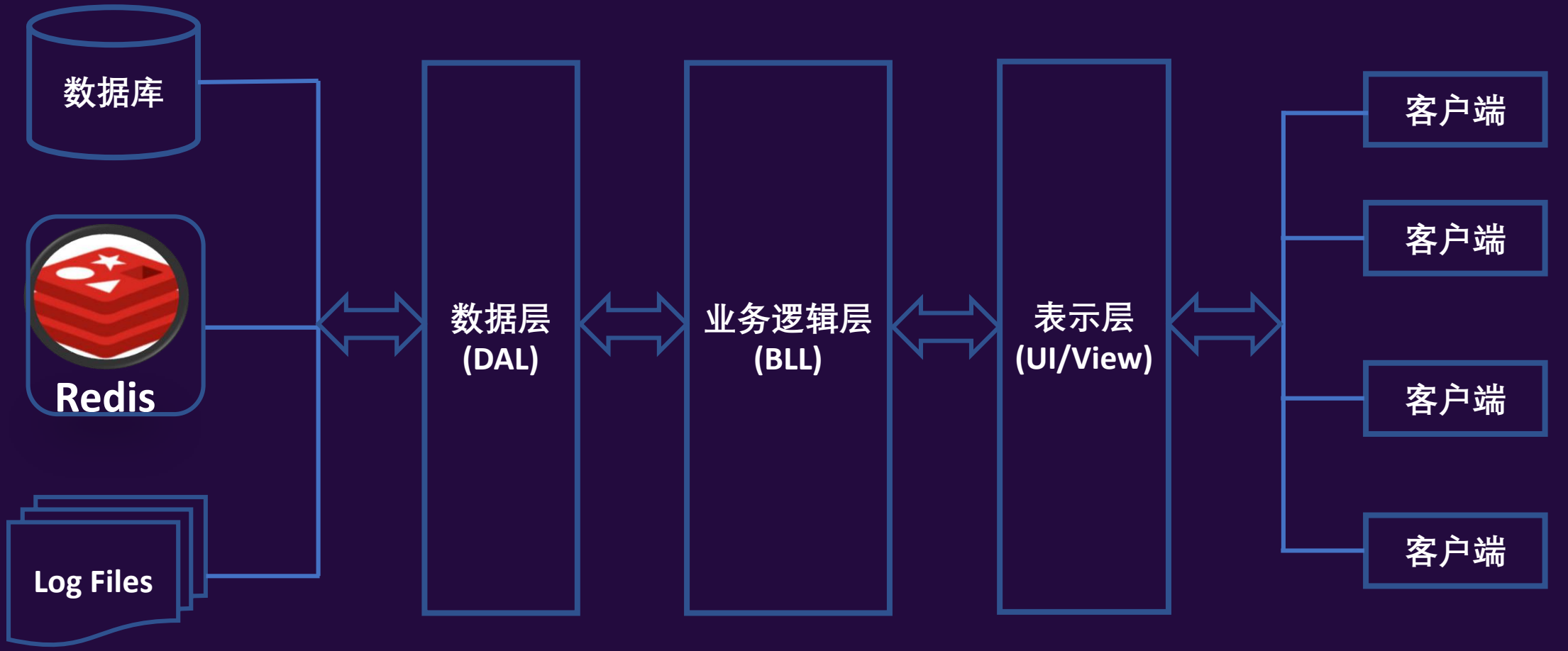
一个真正可以使用的框架，必须是功能完备的

- 操作数据库……DbHelper（ADO.net、EF、Dapper...）
- 记日志……txt/database、log4net/nlog
- 缓存的存取…… memory、memcahe or redis?
- 常用小工具集…… Md5/JsonHelper/DateTimeHelper/…
- NoSql的支持…… MongoDbHelper
- 对依赖的第三方系统的解耦…… 消息队列(RabbitMQ、ZeroMq…)
- …… （暂时还没有想到）

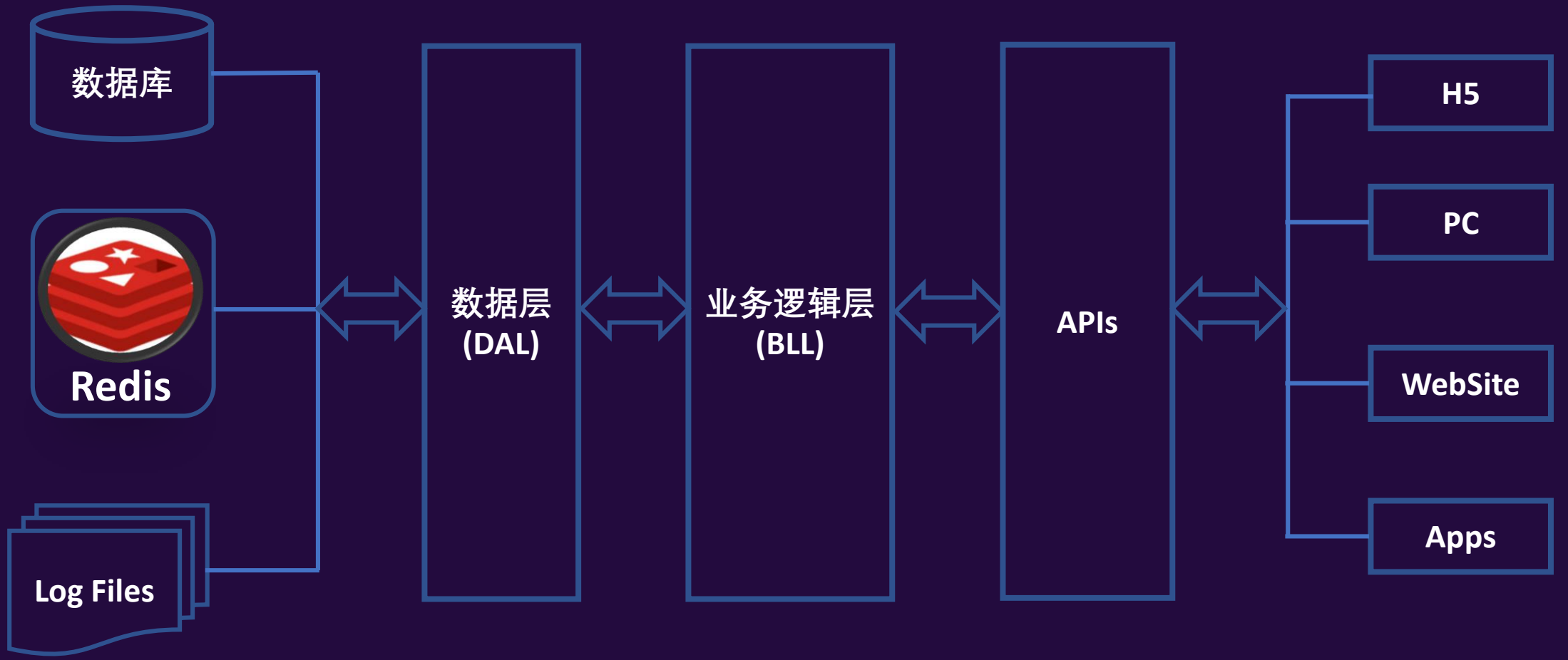


三层架构&画图---一图胜千言

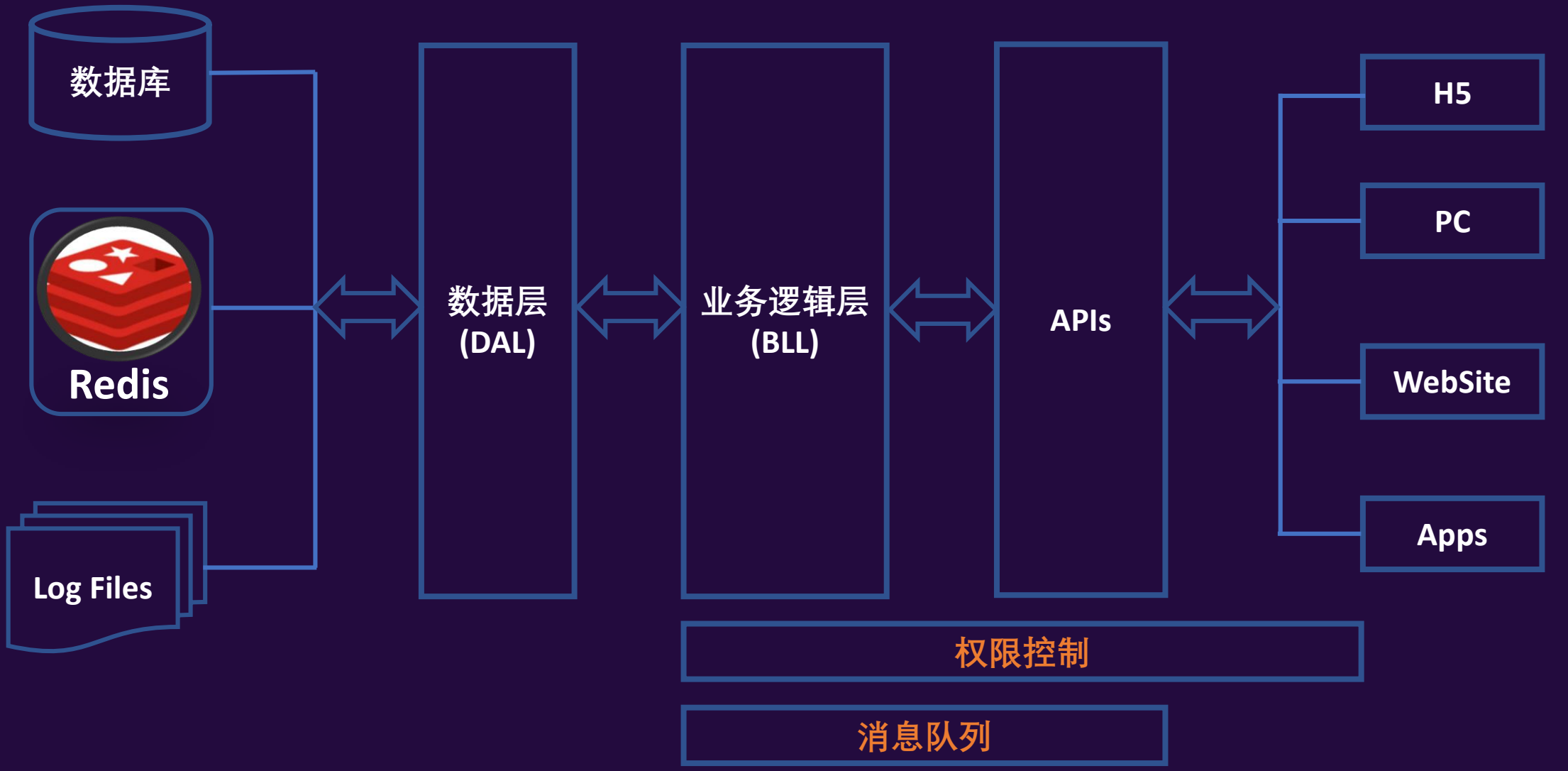
系统架构图—v1.0


数据库
Redis
Log Files
数据层
(DAL)
业务逻辑层
(BLL)
表示层
(UI/View)
客户端
客户端
客户端
客户端

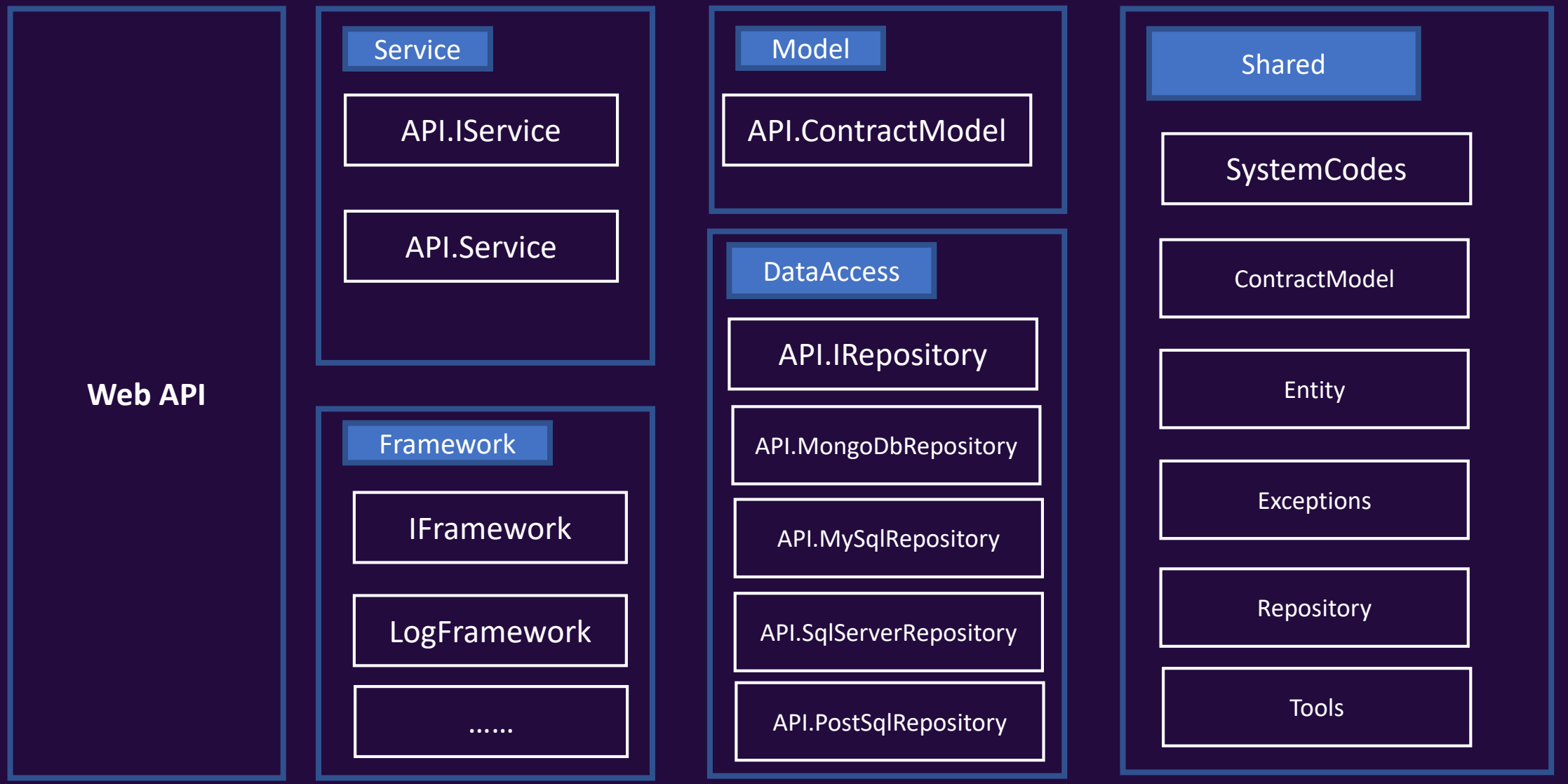
系统架构图--v1.1



系统架构图--v1.2

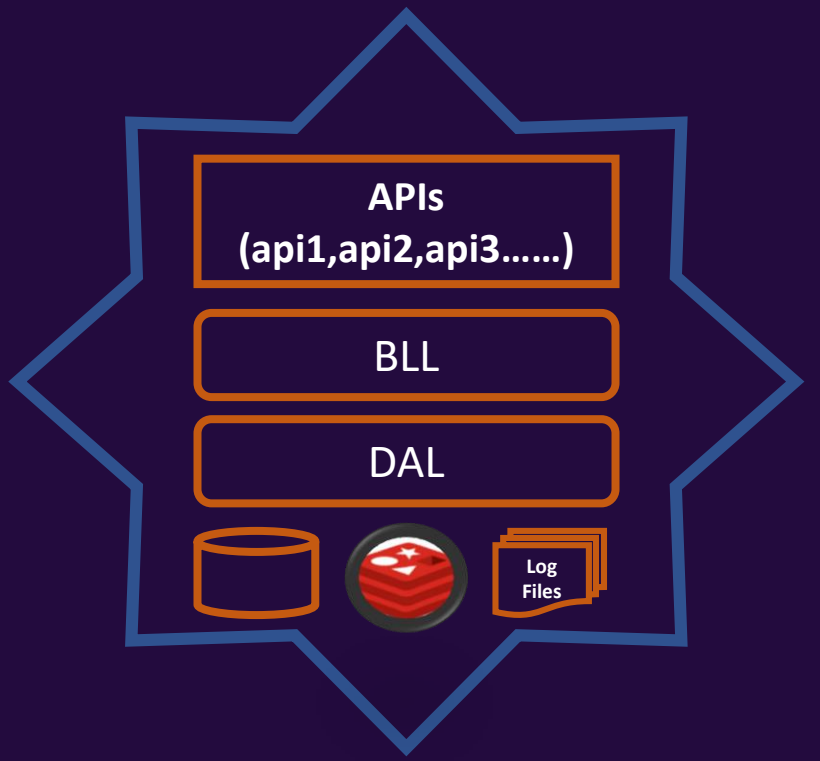

数据库
Redis
Log Files
数据层
(DAL)
业务逻辑层
(BLL)
APIs
H5
PC
WebSite
Apps
权限控制
消息队列

代码结构图

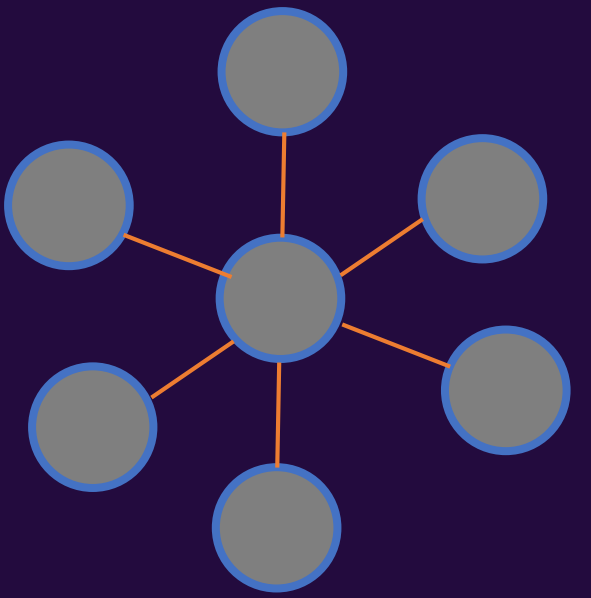

Web API
Service
API.IService
API.Service
Framework
IFramework
LogFramework
……
Model
API.ContractModel
DataAccess
API.IRepository
API.MongoDbRepository
API.MySqlRepository
API.SqlServerRepository
API.PostSqlRepository
Shared
SystemCodes
ContractModel
Entity
Exceptions
Repository
Tools

架构范式的改变

从单体到微服务



单体架构



微服务架构

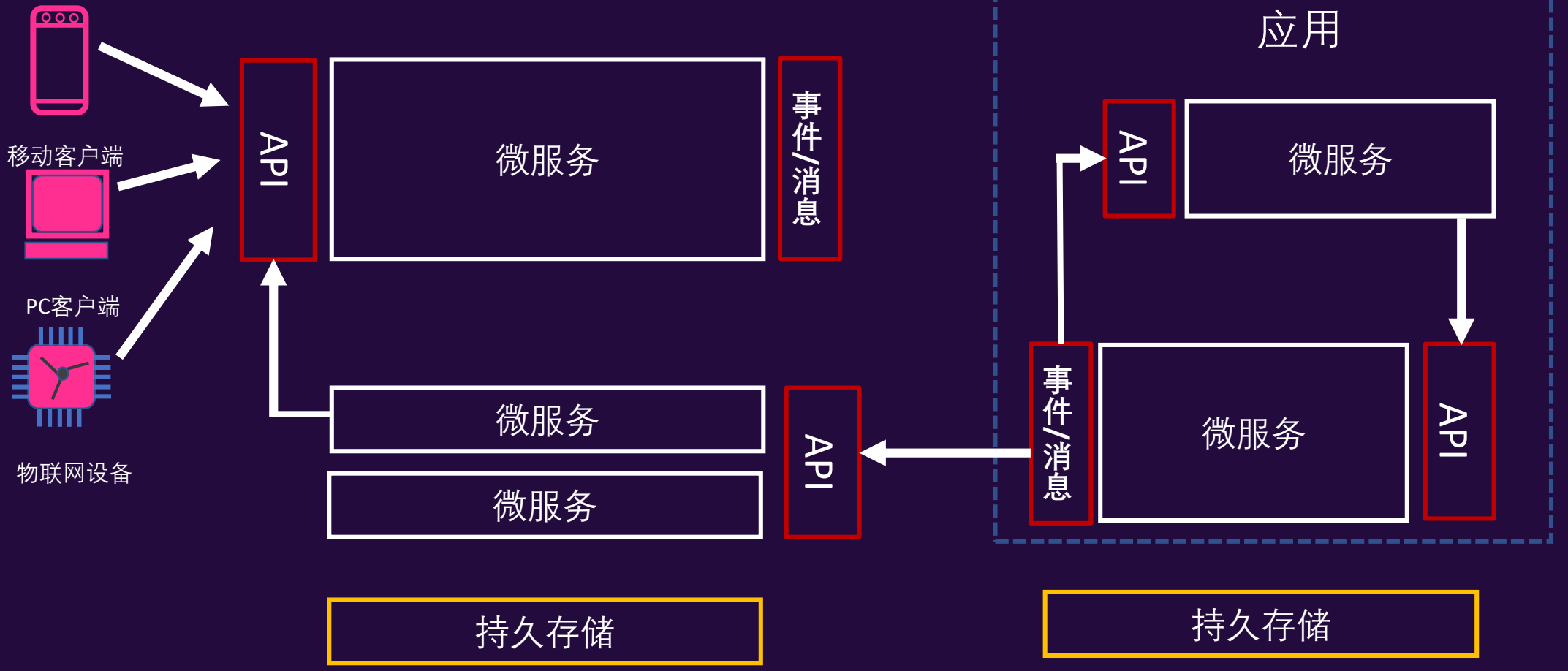
微服务的优势

1. 技术异构性：微服务可以轻松采用不同的技术栈。（C#/Java/PHP/Golang/……）
2. 弹性：一个服务不可用不会导致整个系统不可用。（断胳膊少腿，也死不了）
3. 扩展：可以只针对那些需要扩展的微服务进行扩展。
4. 简化部署：不用重新部署整个应用，只需要部署个别服务，并可以快速回滚。
5. 与组织结构相匹配：符合康威定律（请找度娘）。
6. 可组合性：对已有功能组合实现新的应用。
7. 可替代性：重新实现某个服务相对容易些。

微服务的挑战

1. 版本：各个微服务应该用统一版本号呢，还是各自独立版本？
2. 代码：重复的代码怎么办？（这还用问吗？抽出来，新耦合？……）
3. 界面：服务拆分了，那用户界面怎么办？
4. 数据：各服务是共享数据还是独占数据？（如果共享，这将成为耦合的点？）
5. 分布式：微服务架构是典型的分布式架构。
6. 监控：监控对于微服务架构十分重要。（甩锅必备，死道友莫死贫道！）
7. 安全：服务之间的调用，能信任吗？

系统架构图（微服务）--v2.0



回头思考下，我们的框架应该怎么改，还缺少哪些东西？

- 操作数据库……DbHelper（ADO.net、EF、Dapper...）
- 记日志……txt/database、log4net/nlog
- 缓存的存取…… memory、memcahe or redis?
- 常用小工具集…… Md5/JsonHelper/DateTimeHelper/…
- NoSql的支持…… MongoDbHelper
- 对依赖的第三方系统的解耦…… 消息队列(RabbitMQ、ZeroMq…)
- …… （暂时还没有想到）

回头思考下，我们的框架应该怎么改，还缺少哪些东西？

- 记日志……ELK（Elastic+LogStash+Kibana）
- 微服务的管理（注册和发现）
- 配置信息的管理（配置中心）
- 对调用链条上的API节点进行监控(*甩锅之必备*)
- 容器化
- 灰度发布
- 容错机制&重试机制

回头思考下，我们的框架应该怎么改，还缺少哪些东西？

- 记日志……ELK（ElasticSearch+LogStash+Kibana）
- 微服务的管理（注册和发现） ……Consul(Ocelot)/Eureka(Steeltoe)
- 配置信息的管理（配置中心） ……Apollo（携程的阿波罗）/keydb
- 对调用链条上的API节点进行监控(*APM*) ……ZipKin/Pinpoit/SkyWalking/Cat
- 容器化……Kubernetes(k8s)
- 灰度发布……Kubernetes(k8s)+Istio/自己动手
- 服务间通信&容错机制&重试机制……消息队列是个好东西（RabbitMQ有话说）

从开发框架到开发平台

- 开发框架

  代码脚手架（尽可能的简洁&易用）

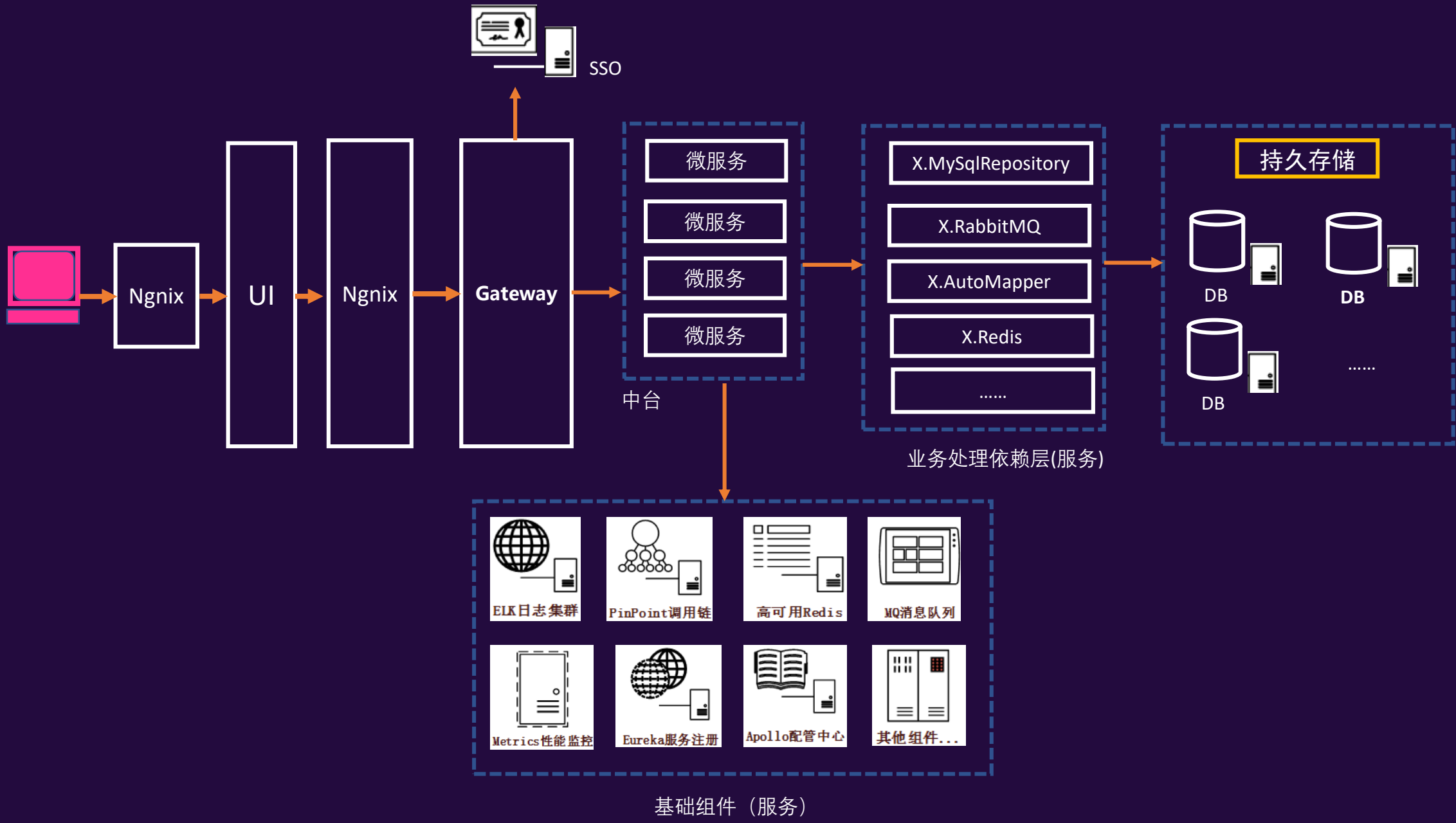
- 武器库

  X.AutoMapper
  X.Configuration.Yaml
  X.CoreBase/Extensions
  X.CoreBase.Metrics
  X.Dapper
  X.Nlog
  X.MongoDB
  X.Pinpoint.Agent
  X.RabbitMQ
  X.Redis
  X.ZipKinClient
  XLog.Kafka
  X.Ocelot
  ……

- 基础设施服务

  ELK日志集群　高可用Redis集群　MQ(Paas)　Metrics Server
  Pinpoint集群　Eureka集群　Apollo配管中心　……

系统架构图--v2.x


SSO
Ngnix
UI
Ngnix
Gateway
微服务
微服务
微服务
微服务
中台
X.MySqlRepository
X.RabbitMQ
X.AutoMapper
X.Redis
.....
业务处理依赖层(服务)
持久存储
DB
DB
DB
.....
ELK日志集群
PinPoint调用链
高可用Redis
MQ消息队列
Metrics性能监控
Eureka服务注册
Apollo配管中心
其他组件...
基础组件（服务）

他山之石

Kratos

Language Go | build passing | godoc reference | go report A+

快速开始

快速使用kratos项目，可以使用`kratos`工具，如下：

```
go get -u github.com/bilibili/kratos/tool/kratos
cd $GOPATH/src
kratos new kratos-demo
```

根据提示可以快速创建项目，如kratos-demo就是通过工具创建生成。目录结构如下：

```
├── CHANGELOG.md            # CHANGELOG
├── CONTRIBUTORS.md         # CONTRIBUTORS
├── README.md               # README
├── api                     # api目录为对外保留的proto文件及生成的pb.go文件，注：需要"--proto"参数
│   ├── api.proto
│   ├── api.pb.go           # 通过go generate生成的pb.go文件
│   └── generate.go
├── cmd                     # cmd目录为main所在
│   └── main.go             # main.go
├── configs                 # configs为配置文件目录
```

小结

- 框架的意义：让开发人员的精力更专注于业务功能的实现；
- 框架的选择：适合的框架才是最好的框架；
- 小小的感悟：遇到好的框架（代码）请不要错过，多多临摹并尝试下微创新；

- 终极目标：搞个自己的CPU，代码想怎么写，就怎么写！

Thank You!


0利率 0负担 终身免费换电免费质保
点击车型，开始探索
es8
es6

ES8体验

ES6体验