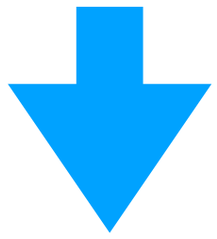
可复用最佳实践-技术篇

梁朝东

如何很好复用已有的?

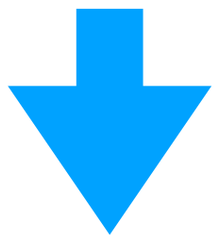
如何让做出的能被好复用?



复用在项目初始就应仔细考虑

问题和痛点

“我们为什么要再造一个轮子？”



“我们怎样才能造好一个轮子？”

轮子的特点

接口标准

功能单一

可单独替换

最佳实践一：接口标准化

- 为什么接口如此重要?
- 什么是好的接口?
  - 功能独立、只做并做好一件事
  - 易于扩展
  - 原子性尽量在一个接口内保证
  - 实现永远不要影响接口

Tips

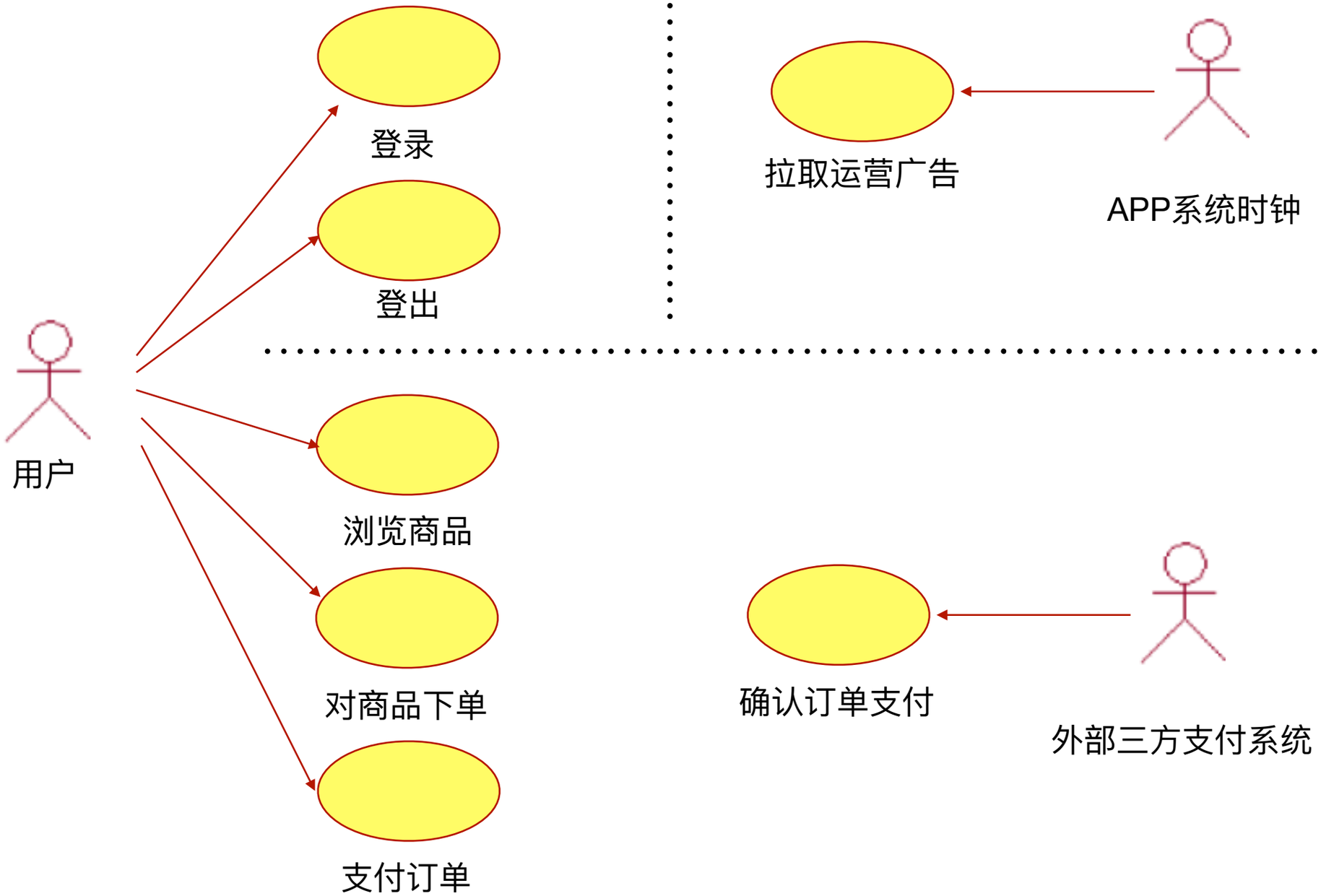
- 接口（接口名和交换的数据）应在开发前期设计好，且文档和定义清晰。需求分析的质量直接影响了接口质量

  接口名建议：动词+名词，如 create_order

- 函数名称清晰，不恰当的函数名称，往往是不恰当设计的征兆

- 只增加，永远不要删除函数与接口

UseCase驱动的接口设计



关键点： 角色（Actor），动作（Action）

由用例确定业务实体对象

业务逻辑驱动

V.S.

业务实体驱动

用基于业务逻辑的用例驱动，用业务实体的CRUD检查接口的完备性

- 红包相关
  - 获取可用兑换码接口（仅测试用需要配置权限）red_get_redeem_code_test
  - 我的红包 red_user_list　　get_user_reds
  - 根据页数查询我的红包 red_user_list_by_page　　可考虑跟上一个合并
  - 根据user_id取得红包列表（后台用） red_user_list_by_user_id　　get_user_reds
  - 我的红包余额 red_user_balance　　get_red_balance
  - 查看红包信息 get_red_info
  - 查看红包消费记录 get_red_info_detail　　get_red_info_details
  - 兑换兑换码 red_exchange_redeem_code　　exchange_red_redeem_code
  - 支付用红包列表 red_user_list_for_pay　　get_reds_for_pay
  - 派发红包（购买相关） red_distribute　　distribute_reds
  - 使用红包（全额） red_user_cost_all　　pay_order_by_red
  - 使用红包（组合支付） red_user_cost_part　　pay_order_by_red_mix
  - 红包过期提醒 red_expires　　send_red_expire_notices
  - 红包开通 red_big_initial　　register_user_red

最佳实践二

架构框架化

功能插件化

最佳实践：架构框架化

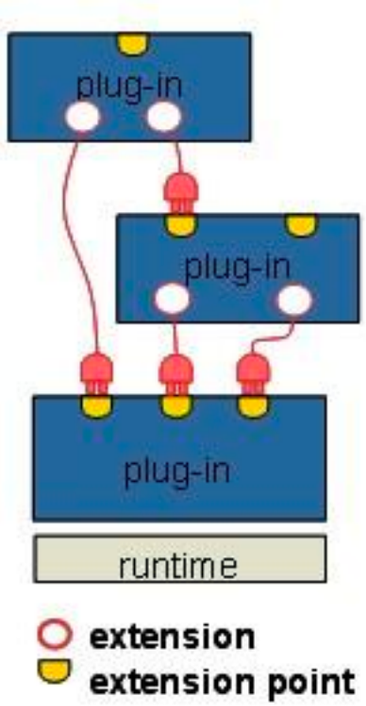
- 什么是框架?

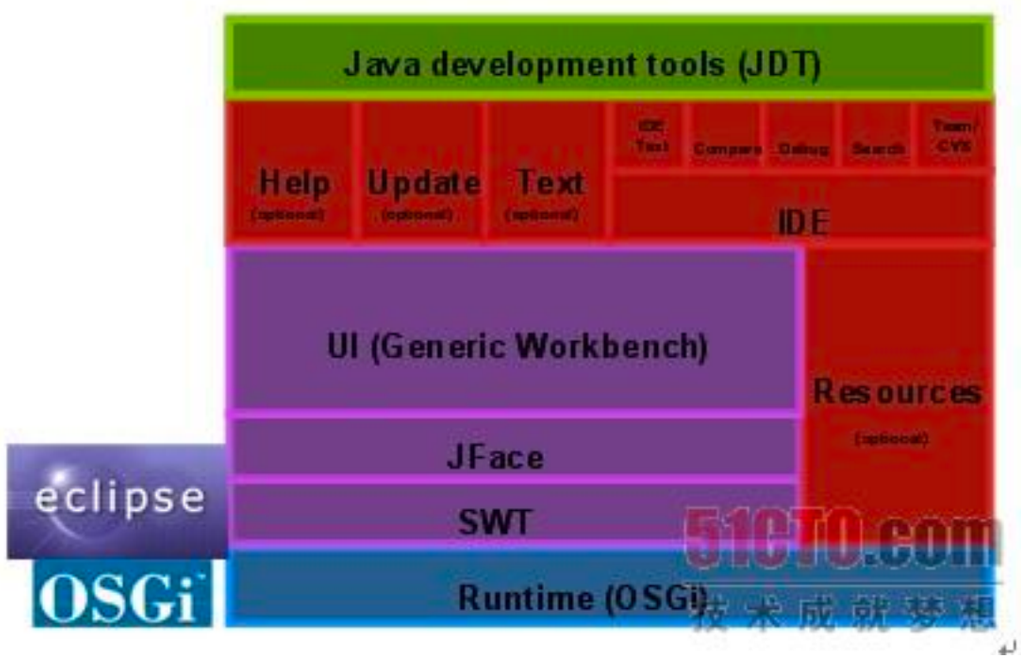
  基础的东西都有，并且是**可扩展**的

- 例如：Eclipse - 即插即用的总线OSGI Impl，和很多可扩展的“槽点” Plugin Extension Points)

最佳实践：功能组件化

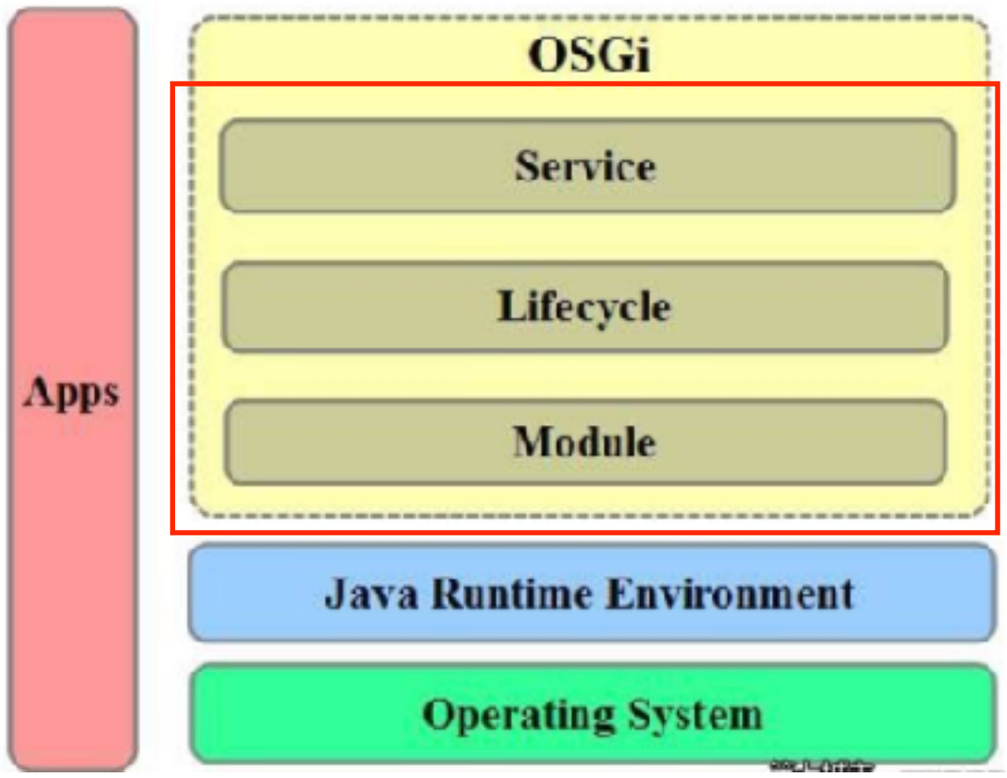
- 基于扩展点（extension points）
- 扩展点实现还可引入新扩展点





Plugins

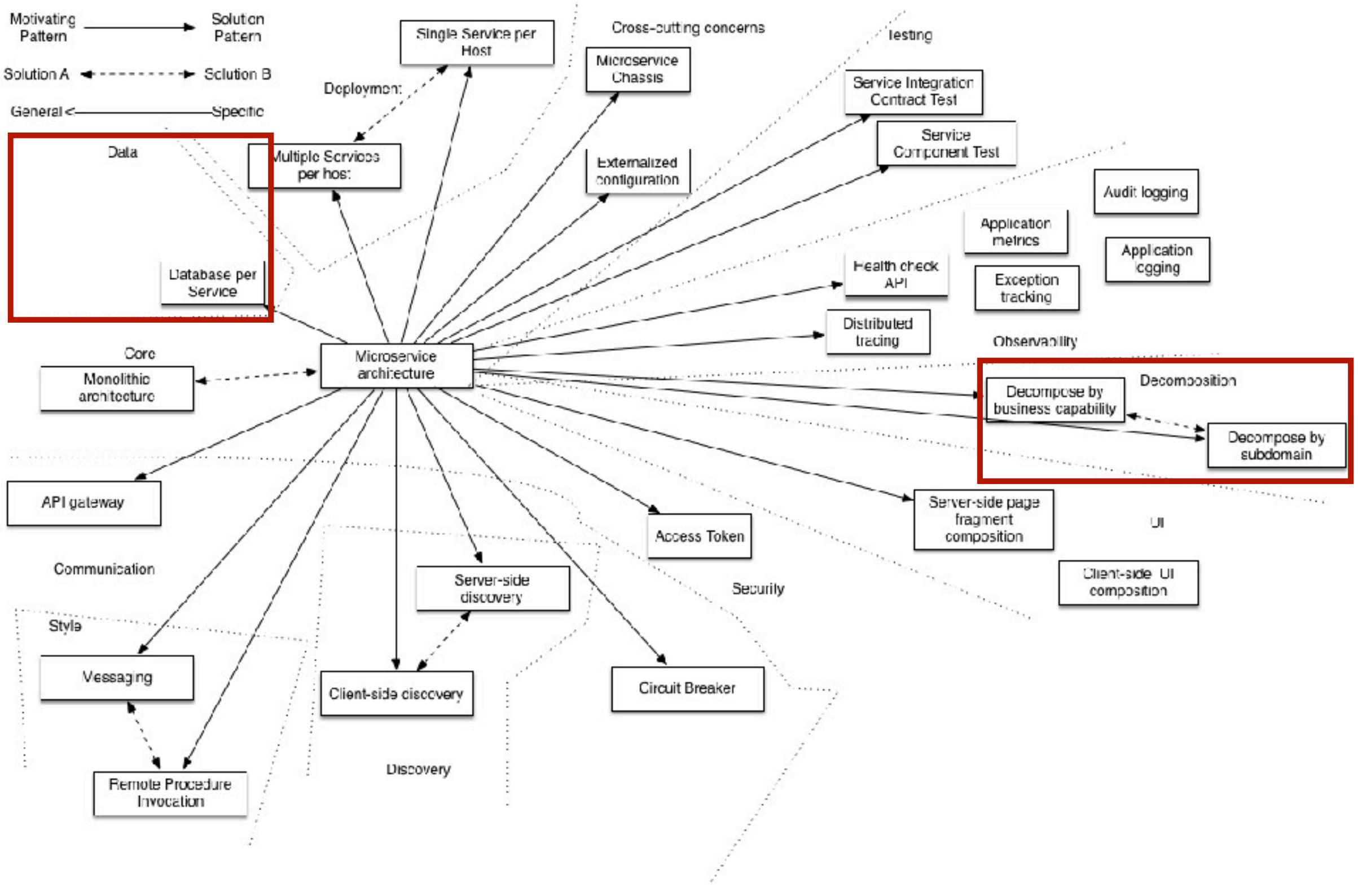
- Bundle : **组件封装问题**
- Lifecycle： Initialize， Startup, Shutdown
- Service：服务功能实现



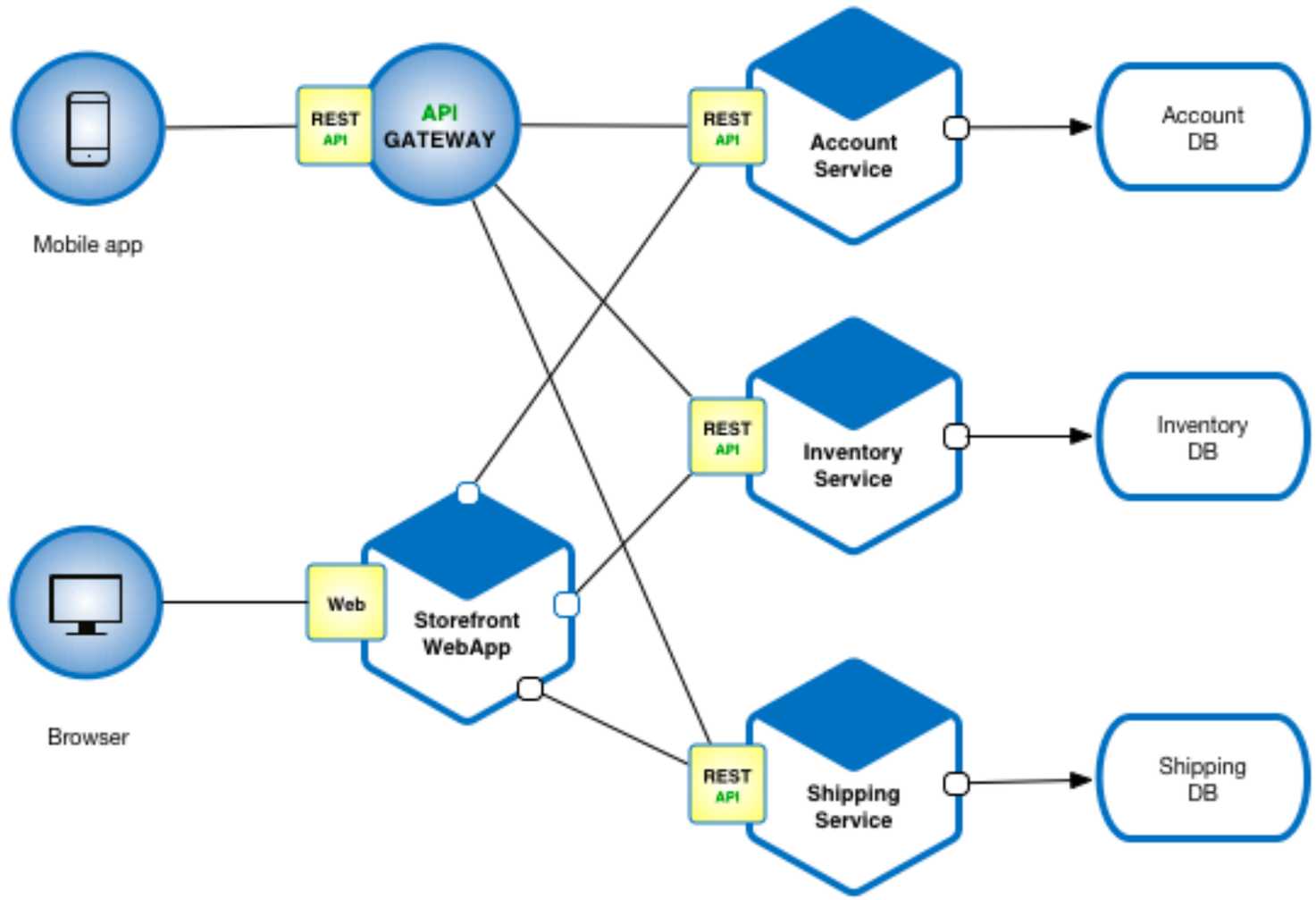
代码展示

最佳实践三：后台功能微服务化

微服务相关模式


Motivating Pattern
Solution Pattern
Solution A
Solution B
General
Specific
Single Service per Host
Deploymen:
Multiple Services per hos:
Data
Database per Service
Cross-cutting concerns
Microservice Chassis
Externalized configuration
Testing
Service Integration Contract Test
Service Component Test
Audit logging
Application metrics
Application logging
Exception tracking
Health check API
Distributed tracing
Observability
Core
Monolithic architecture
Microservice architecture
Decomposition
Decompose by business capability
Decompose by subdomain
API gateway
Server-side page fragment composition
UI
Client-side UI composition
Access Token
Security
Communication
Server-side discovery
Style
Messaging
Client-side discovery
Circuit Breaker
Discovery
Remote Procedure Invocation

解耦应用到微服务



根据业务实体名词

根据业务实体名词

根据业务能力（动）名词

基本原则一

对扩展开放，对修改关闭

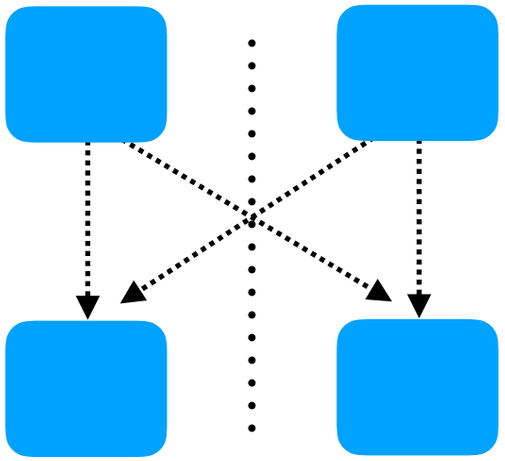
How

标准且可扩展的接口

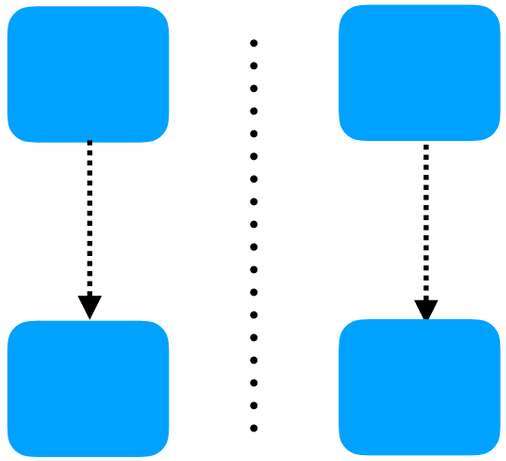
Plugin设计可扩展点

基本原则二

弱依赖，强隔离



解耦



✗

How

√

组件化(Plugin Module)

服务间：微服务

其他：如 消息队列

总结

三个最佳实践

两大基本原则

设计好接口

做好框架和扩展

做好垂直的微服务

思考

- 设计一个会员系统，思考：从哪扩展、扩展方式是什么？怎样才可使它可被复用？

Q&A