紫川

宋小菜架构师

《宋小菜技术的领域驱动设计实践》

GraphQL Party | 杭州

宋小菜技术的
领域驱动设计 (DDD) 实践

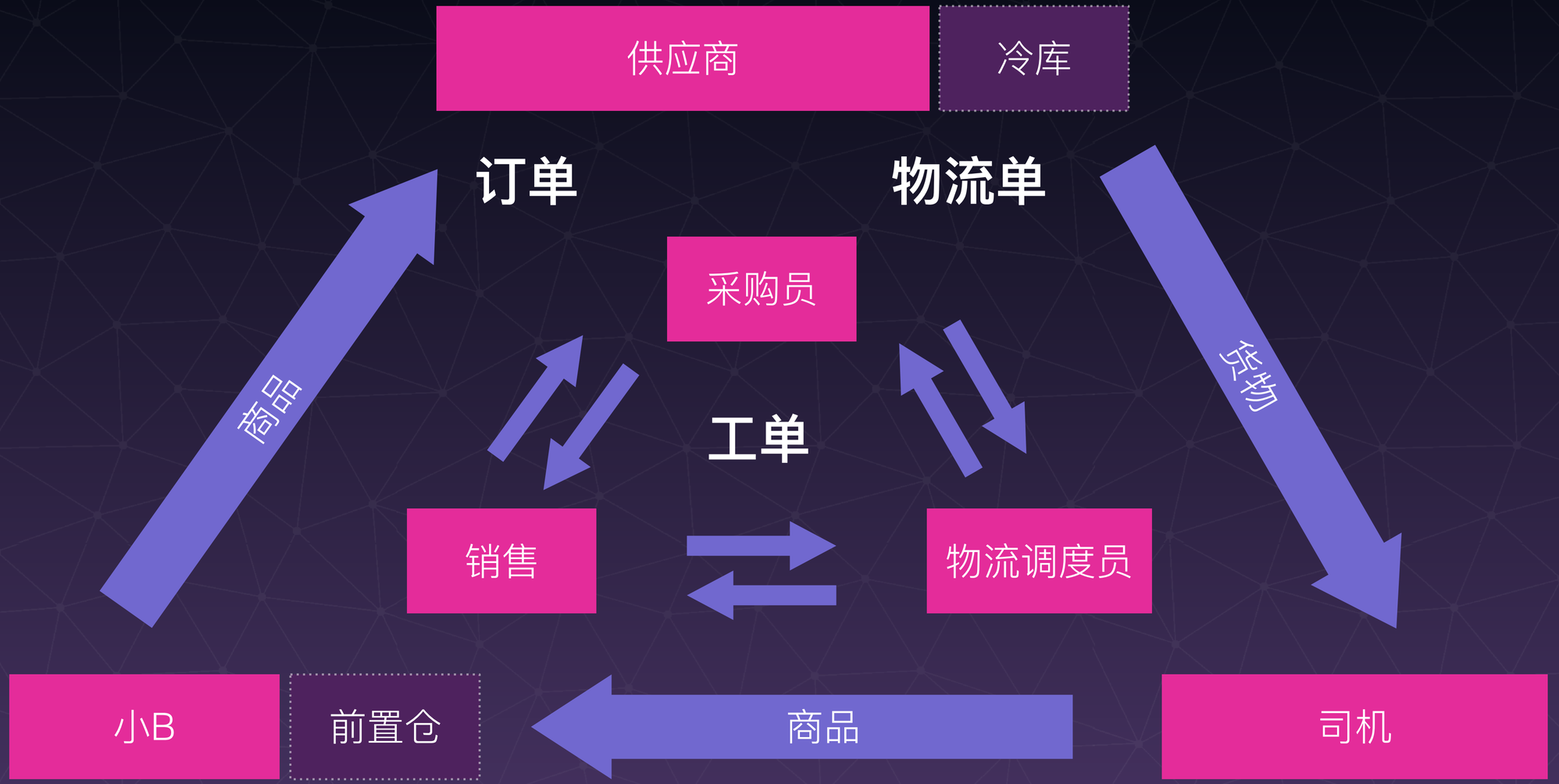
宋小菜 | 紫川

宋小菜 SONGXIAOCAI.COM X CODING CLOUD DEVELOPMENT

DDD 保证和提升 GraphQL 实施的价值

业务背景

- 业务复杂多变
- 业务链路长
- 服务角色多


供应商
冷库
订单
物流单
采购员
商品
工单
货物
销售
物流调度员
小B
前置仓
商品
司机

技术背景

- 前后端协同效率提升出现瓶颈
- 前后端对业务的认知出现分歧

宋小菜老架构


宋小菜
宋小福
采秘
宋小菜司机
宋小菜-供应商
网关框架应用
网关平台
宋小菜APP页面接口
宋小福APP页面接口
采秘APP页面接口
麦大蔬APP页面接口
登录服务
互动服务
消息服务
订单服务
物流调度服务
直发协调服务
营销服务
订单中心(OC)
商品中心(IC)
库存中心(SC)
用户中心UC
价格中心(PC)
营销中心(AC)
OSS
RDS

宋小菜产品开发流程


前后端协同阻塞节点
需求讨论
产品prd
评审
UI评审
系统设计
接口设计
接口
mock
代码开发
单元测试
集成测试
产品验收
灰度发布
产品上线

页面驱动设计

技术设计依赖UI设计

- 一个页面对应一个后端接口,交互和视觉调整带来前后端整体调整。
- 前后端开发被UI捆绑,不能独立并行开发。
- 后端开发了大量胶水代码来组装页面,浪费了 25% 的开发工作。
- 前后端对业务理解不统一,在集成测试阶段需要花大量时间来沟通,最后使用畸形方案来妥协。

服务端累计产生了 111 个面向页面的类 →

页面驱动设计

无法满足业务发展要求

宋小菜新架构

- 基于 GraphQL 实施改变协同方式
  - 省去后端胶水代码开发,后端更专注业务的理解和抽象。
  - 前后端分离,前端可以独立参与产品功能开发,复用基于 GraphQL 封装的页面组件。

GraphQL+页面驱动设计能解决问题么？

- 没有抽象和稳定的后端接口,接口随着页面设计变化不断调整。
- 一个页面对应一个后端接口,不同的开发同学会开发出重复的后端接口。
- 后端接口泛滥, GraphQL 无法对接确定的数据模型。

GraphQL 实施

需要确定的数据模型

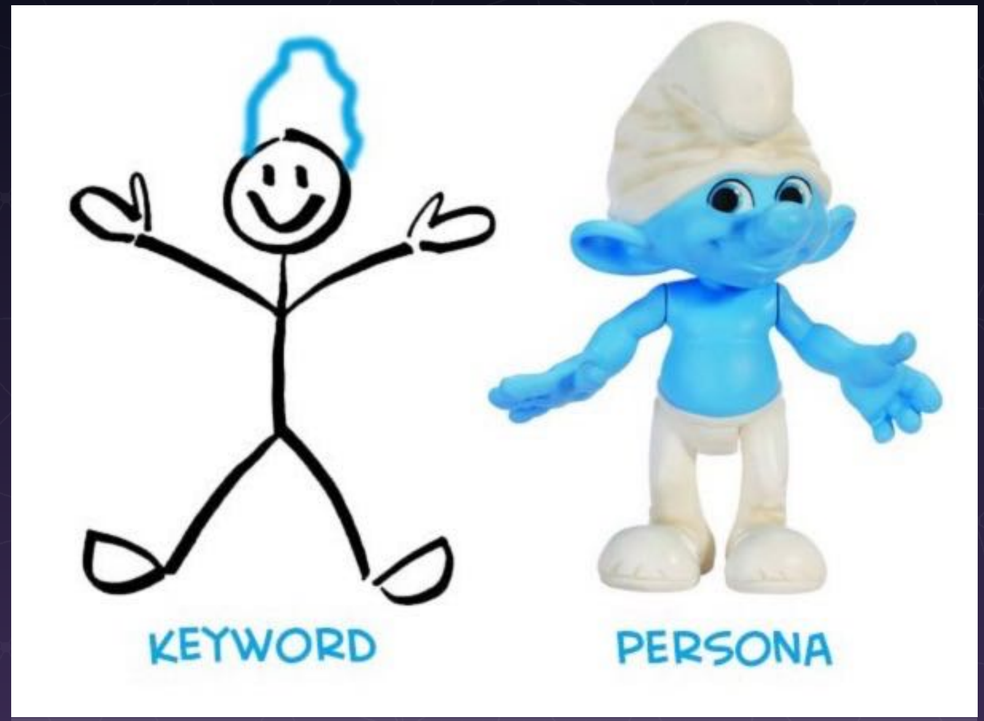
领域驱动设计

可以提供确定的数据模型

领域驱动设计 (DDD) 是什么？

一套系统建模的方法论

- 领域驱动设计可以合理的进行服务划分和治理,避免系统设计上的业务耦合。
- 领域模型准确表达业务语言,让技术同学更容易理解业务需求。
- 领域模型抽象和定义业务元素,促进沟通的准确性,实施的确定性。



建模就是对现实事物的抽象和定义

DDD 和 GraphQL 结合实施的价值

- 通过 GraphQL 的实时聚合数据模型,可以快速支持 App 的页面展示，充分发挥领域模型的价值。
- DDD 提供了在具体业务场景下确定性的模型和接口，减少 GraphQL 的 schema 的重复定义和服务接口的重复调用。
- 统一前后端同学对于业务的认知，基于统一的业务模型沟通和交流，避免了对需求理解不一致带来的额外沟通成本。

宋小菜
如何实施领域驱动设计 (DDD)?

业务需求	产品方案	技术方案	工程落地
- 确定业务问题边界(确定需求是那块业务域下) - 确定业务场景 - 确定业务术语	- 确定产品用例 - 根据产品用例中的业务术语抽象领域模型	- 在正确的业务域的系统中设计领域模型 - 使用产品用例验证领域模型	- 基于 DDD 搭建规范的工程结构 - 基于 DDD 创建规范的包和类成员

工程模块管理规范

- 系统部署
  - 一个大的业务域一个系统,独立部署
- 启动工程
  - 基于 SpringBoot 搭建
  - 只包含容器相关能力（如：SpringBoot启动配置，子域扫描配置，不允许出现业务代码）
- 工程模块
  - 大的业务域下的子域单独一个工程模块管理,包含实现和接口 2个子工程
  - 子域之间不允许相互依赖 core 层,如有需求可以通过依赖 client 通过 RPC 通讯.
  - 工程模块包含业务的规模越来越大,可以分拆独立部署

用户中心工程结构

工程模块的包规范

- 应用层
  - 暴露服务能力
  - 隔离领域层,防止业务泄漏
- 领域层
  - 沉淀和保护领域模型
  - 管理业务域的业务规则,纯粹的业务表达,不和具体的技术框架耦合
- 基础设施层
  - 管理工程中技术框架
  - 不包含业务规则和业务描述

DDD 实施建议

- 避免教条化实施,特别使用 SSM 套件的团队，DDD 和 Spring 框架有冲突的地方,需要权衡和取舍。
- DDD 不可能让团队中所有同学都理解,让团队中开发经验丰富的同学进行设计和实施。
- DDD 是一套方法论,在实施之前,需要逻辑完备的落地规范.在实施过程中,需要不断优化规范。

Q&A