



# 给我做一个Java微服务项目

Week #2 Java与微服务 / 刘俊强



欢迎关注StuQ公众号



欢迎关注我的微信公众号

# 大纲

- 依赖关系工具
- Java常用微服务框架
- 如何设计微服务

# 依赖关系工具

又称 构建系统

# 常用依赖关系工具/构建系统

- Apache Ant
- Apache Maven
- Gradle

# Apache Ant

- 基于XML, project、target、task
- 最早被Java领域广泛使用的构建系统
- 配置过于复杂, 简单任务的工作量都很大
- 项目没有标准的结构约定



```
<?xml version="1.0" encoding="ISO-8859-1"?>
<project name="MyTask" basedir="." default="jar">

    <property name="src.dir" value="src"/>
    <property name="classes.dir" value="classes"/>

    <target name="clean" description="Delete all generated files">
        <delete dir="${classes.dir}" failonerror="false"/>
        <delete file="${ant.project.name}.jar"/>
    </target>

    <target name="compile" description="Compiles the Task">
        <mkdir dir="${classes.dir}"/>
        <javac srcdir="${src.dir}" destdir="${classes.dir}"/>
    </target>

    <target name="jar" description="JARs the Task" depends="compile">
        <jar destfile="${ant.project.name}.jar" basedir="${classes.dir}"/>
    </target>

</project>
```

# Apache Maven

- 依赖关系工具、构建系统
- 主要通过XML进行配置，pom.xml，可以实现：
- 打包、依赖、代码库、发布、文档、测试报告等
- 项目的生命周期管理
- 简化了项目开发，重用性、一致性都有提高

```

<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>io.junq.examples.emall.boot</groupId>
    <artifactId>account-service</artifactId>
    <version>1.0</version>
    <packaging>jar</packaging>

    <name>account-service</name>
    <description>account microservice for eMall</description>

    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>1.5.1.RELEASE</version>
        <relativePath /> <!-- lookup parent from repository -->
    </parent>

    <properties>

```

# Gradle

- 下一代依赖关系工具、构建系统
- 语法简洁，基于Groovy的DSL，对XML无爱
- 保持Ant的自由度，低于Maven的学习门槛
- 后台守护进程，减少构建时间
- 支持 Ant、Maven集成

```

buildscript {
    repositories {
        mavenCentral()
    }
    dependencies {
        classpath("org.springframework.boot:spring-boot-gradle-plugin:1.5.1.RELEASE")
    }
}

apply plugin: 'java'
apply plugin: 'eclipse'
apply plugin: 'idea'
apply plugin: 'org.springframework.boot'

jar {
    baseName = 'gs-spring-boot'
    version = '0.1.0'
}

repositories {
    mavenCentral()
}

sourceCompatibility = 1.8
targetCompatibility = 1.8

dependencies {
    // tag::jetty[]
    compile("org.springframework.boot:spring-boot-starter-web") {
        exclude module: "spring-boot-starter-tomcat"
    }
    compile("org.springframework.boot:spring-boot-starter-jetty")
    // end::jetty[]
    // tag::actuator[]
    compile("org.springframework.boot:spring-boot-starter-actuator")
    // end::actuator[]
    testCompile("junit:junit")
}

```

# Java微服务常用框架

# 微服务相关工具

- 构建系统：Maven、Gradle等
- IDE：Spring Tool Suite、Eclipse、IntelliJ IDEA
- 框架：Spring Boot、Dropwizard等




















# Spring Boot 简介






- 基于Spring依赖注入框架
- 用来简化新Spring应用的初始搭建以及开发过程
- 提供一系列大型项目常用的非功能性特征，比如：  
内嵌服务器，安全，指标，健康检测，外部化配置
- 绝对没有代码生成，也不需要XML配置



# Spring Boot 特性

- 核心特性：SpringApplication、外部化配置、Profiles、日志
- Web应用：内嵌容器、MVC
- 数据库：RDMS、NoSQL、内嵌
- 消息：JMS、AMQP、REST API调用
- 发布就绪：JMX、健康检测、度量指标、命令

- ▼  account-service [boot]
  - ▼  src/main/java
    - ▼  io.junq.examples.emall.boot.account
      - ▶  AccountServiceApplication.java
      - ▶  SwaggerConfiguration.java
    - ▼  io.junq.examples.emall.boot.account.api
      - ▶  ApiError.java
      - ▶  EmailRestExceptionHandler.java
    - ▼  io.junq.examples.emall.boot.account.domain
      - ▶  AccountRepository.java
      - ▶  User.java
    - ▼  io.junq.examples.emall.boot.account.service
      - ▶  AccountService.java
      - ▶  AccountServiceImpl.java
    - ▼  io.junq.examples.emall.boot.account.web
      - ▶  UserRestController.java
  - ▼  src/main/resources
    - ▶  application.yml
  - ▶  src/test/java

- ▼  account-service [boot]
  - ▼  src/main/java
    - ▼  io.junq.examples.emall.boot.account
      - ▶  AccountServiceApplication.java
      - ▶  SwaggerConfiguration.java
    - ▼  io.junq.examples.emall.boot.account.api
      - ▶  ApiError.java
      - ▶  EmailRestExceptionHandler.java
    - ▼  io.junq.examples.emall.boot.account.domain << 数据模型对象及数据访问
      - ▶  AccountRepository.java
      - ▶  User.java
    - ▼  io.junq.examples.emall.boot.account.service << Service
      - ▶  AccountService.java
      - ▶  AccountServiceImpl.java
    - ▼  io.junq.examples.emall.boot.account.web << REST API
      - ▶  UserRestController.java
  - ▼  src/main/resources << 配置文件
    - ▶  application.yml
  - ▶  src/test/java

# 外部化配置 application.yml

---

```
server:
  port: 9999

logging:
  level:
    root: INFO
    io.junq: DEBUG

spring:
  datasource:
    url: jdbc:mysql://localhost:3306/email
    username: email
    password: emailPwd
    driver-class-name: com.mysql.jdbc.Driver
```

# 数据库访问

- JdbcTemplate会被自动配置，可直接 @Autowire
- JPA：Hibernate、Spring Data JPA、Spring  
ORMs
- h2database：支持JDBC API的内嵌数据库
- NoSQL：Spring Data Redis/MongoDB/Solr等













# Dropwizard 简介

- 框架早于Spring Boot，由Yammer团队贡献
- 可快速构建发布就绪的微服务框架
- 提供一系列常用组件，比如健康检测、度量统计、  
内嵌容器Jetty、REST实现、JSON序列化等
- 开箱即用，对XML无爱



# Dropwizard 特性

- 核心：Application、外部化配置、日志
- Web应用：内嵌容器、Jersey
- 数据库：RDMS、NoSQL
- 消息：REST API调用
- 发布就绪：JMX、健康检测、度量指标、管理指令

- ▼  hello-dropwizard
  - ▼  src/main/java
    - ▼  io.junq.examples.dropwizard
      - ▶  HelloDropwizardApplication.java
      - ▶  HelloDropwizardConfiguration.java
      -  io.junq.examples.dropwizard.api
      -  io.junq.examples.dropwizard.core
      -  io.junq.examples.dropwizard.db
      -  io.junq.examples.dropwizard.health
    - ▼  io.junq.examples.dropwizard.resources
      - ▶  HelloDropwizardResource.java
    -  io.junq.examples.dropwizard.service





# 外部化配置 application.yml

```
# service name
serviceName: Hello Dropwizard Service

server:
  requestLog:
    timeZone: UTC
    appenders:
      - type: console
      - type: file
        currentLogFilename: logs/hello-dropwizard-request/dropwizard-request.
        threshold: ALL
        maxFileSize: 100MB
        archive: true
        archivedLogFilenamePattern: logs/hello-dropwizard-request/dropwizard-
        archivedFileCount: 30
        logFormat: "[%d{yyyy-MM-dd HH:mm:ss.SSS}] [%-5level] [%t] [%logger{12
applicationConnectors:
  - type: http
    port: 9979
adminConnectors:
  - type: http
    port: 9978
```

# 数据库访问

- 可集成Spring JdbcTemplate
- 关系型数据库：JDBI
- NoSQL：官方库

# 如何设计微服务？

**API! API! API!**

# Paths

/users/{user-id}



GET /users/{user-id}

用户

## Summary

根据用户id获取用户信息

## Description

通过用户id获取用户对象，此处的用户id对应的是display\_id，真实用户id和password字段不返回真实数据。

## Parameters

Name	Located in	Description	Required	Schema
user-id	path	用户唯一id	Yes	⇔ integer (int32)

## Responses

Code	Description	Schema
200	返回 产品 对象	⇔ { data: ▶ User { } }

**对你的API使用者来说，你的  
API设计就是一份契约。**

# API 设计

- 由一系列能够提供给开发者的能力组成
- 将产品设计与软件开发粘合起来
- 代表了对产品的远景



**微服务拆分设计的基石是API  
设计。**

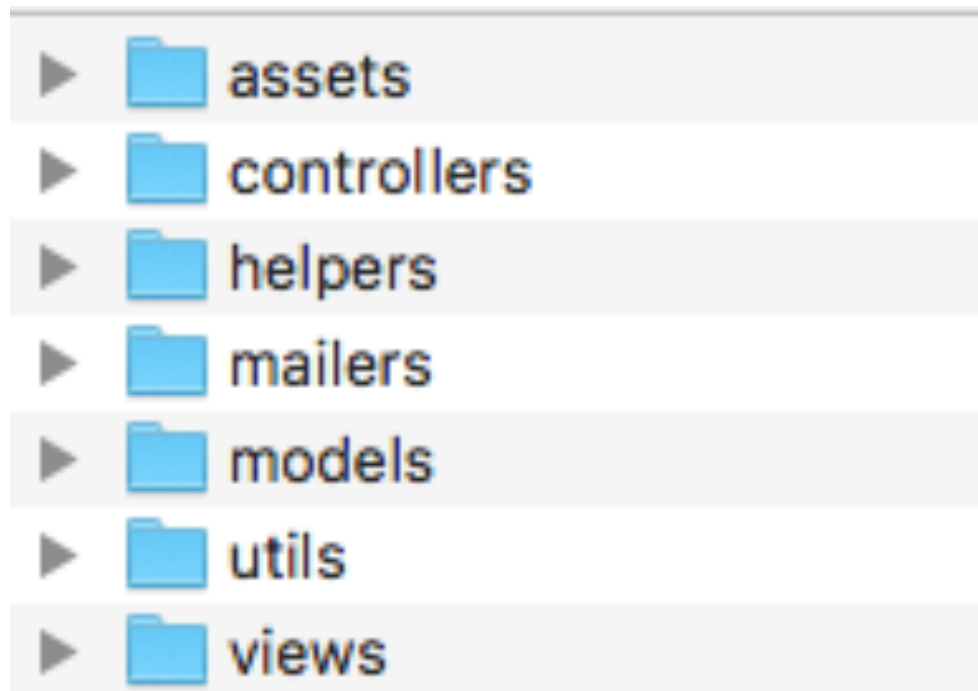
行为	步骤	参与人	描述
下订单	商品清单	消费者	通过搜索或频道来 查阅商品清单
下订单	添加商品到购物车	消费者	添加一件或多件商 品到购物车
下订单	从购物车里面删除 商品	消费者	从购物车里面将一 件或多件商品删除
下订单	清空购物车	消费者	将购物车里面所有 商品删除
下订单	查看购物车	消费者	查看购物车的商品 清单及金额
下订单	结账	消费者	通过购物车的商品 来创建一个订单

**领域驱动设计可以帮助我们  
定义复杂API的上下文边界。**

# 领域驱动设计 Domain Driven Design

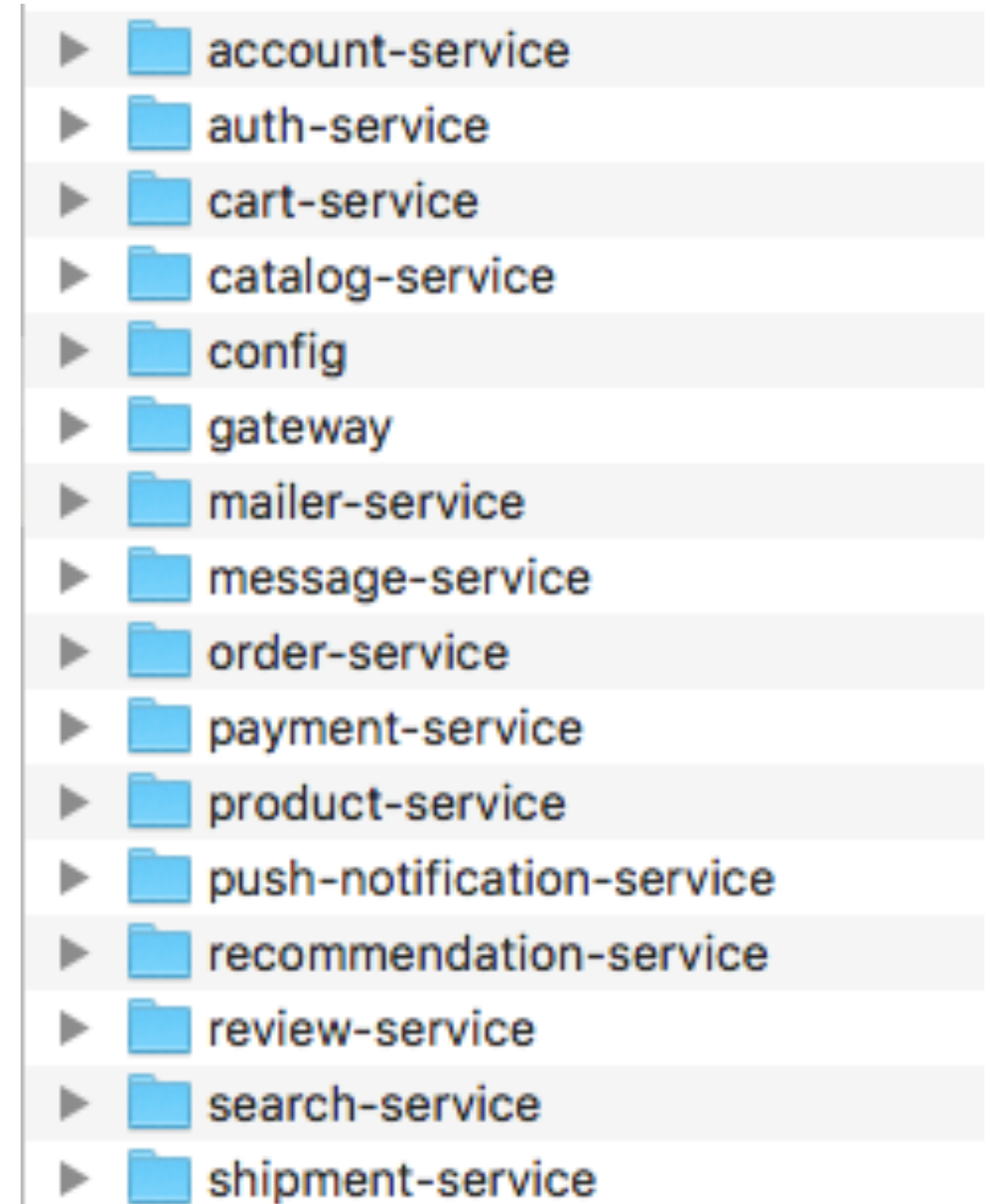
- 领域 (Domain) 是一类认识或行动的区域
- 同一语言 (Ubiquitous Language)
- 有界上下文 (Bounded contexts)
- 来自于 Eric Evans 的书《领域驱动设计》2006年，中文版初版

# 基于功能

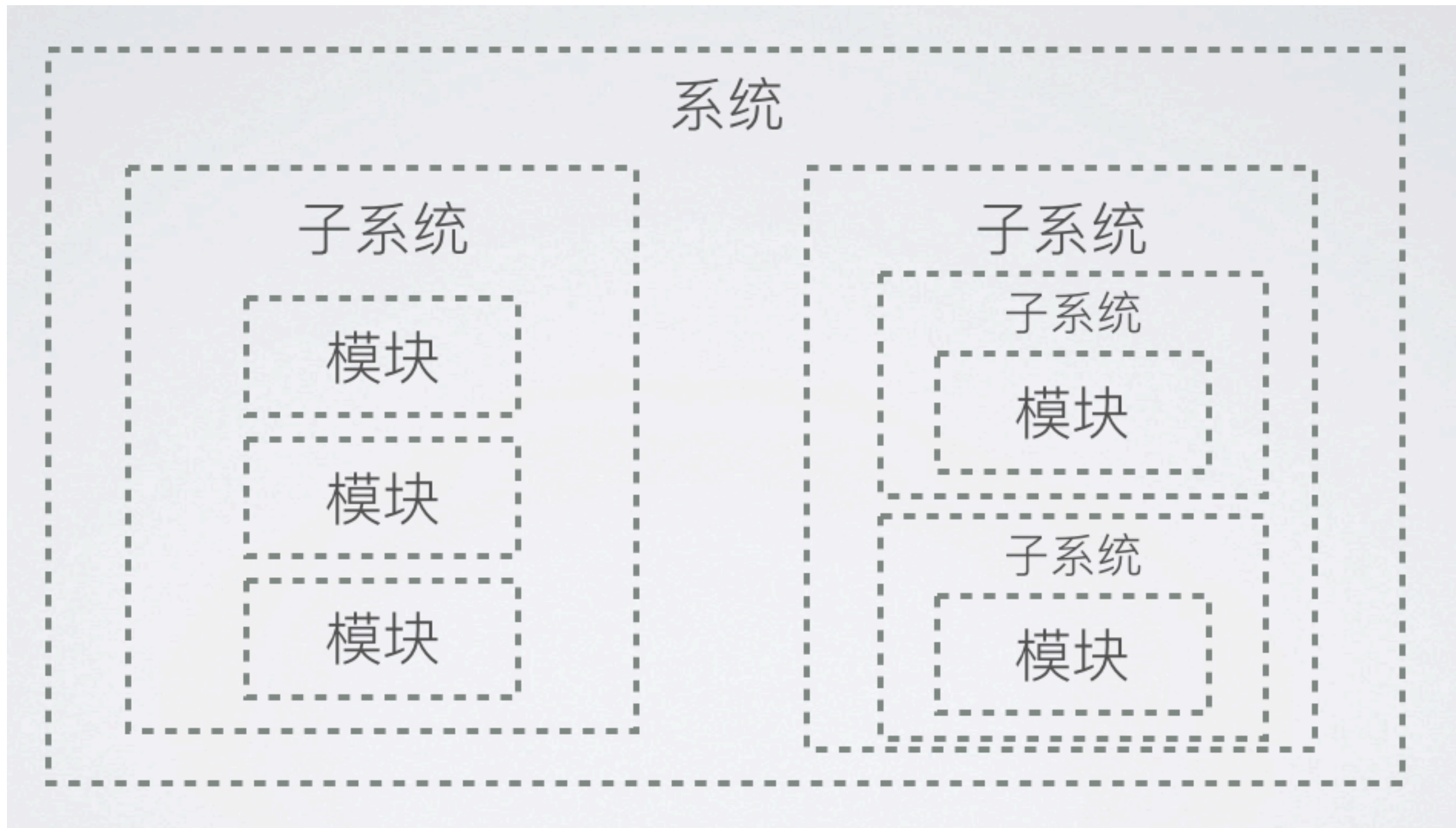


VS.


# 基于模块



# 系统建模






The image shows a LEGO City Police Station set (60047) and its various components. The main box is on the left, featuring a large illustration of the completed station. To the right, the assembled station is shown, a multi-story building with a control room, a helicopter landing pad, and a police car. In the foreground, several smaller boxes for sub-themes like 'City' and 'Police' are visible, along with loose LEGO bricks and minifigures. Three semi-transparent text boxes are overlaid on the image: '系统' (System) on the top left, '模块' (Module) in the center, and '子系统' (Subsystem) at the bottom.

系统

模块

子系统





系统

系统

系统

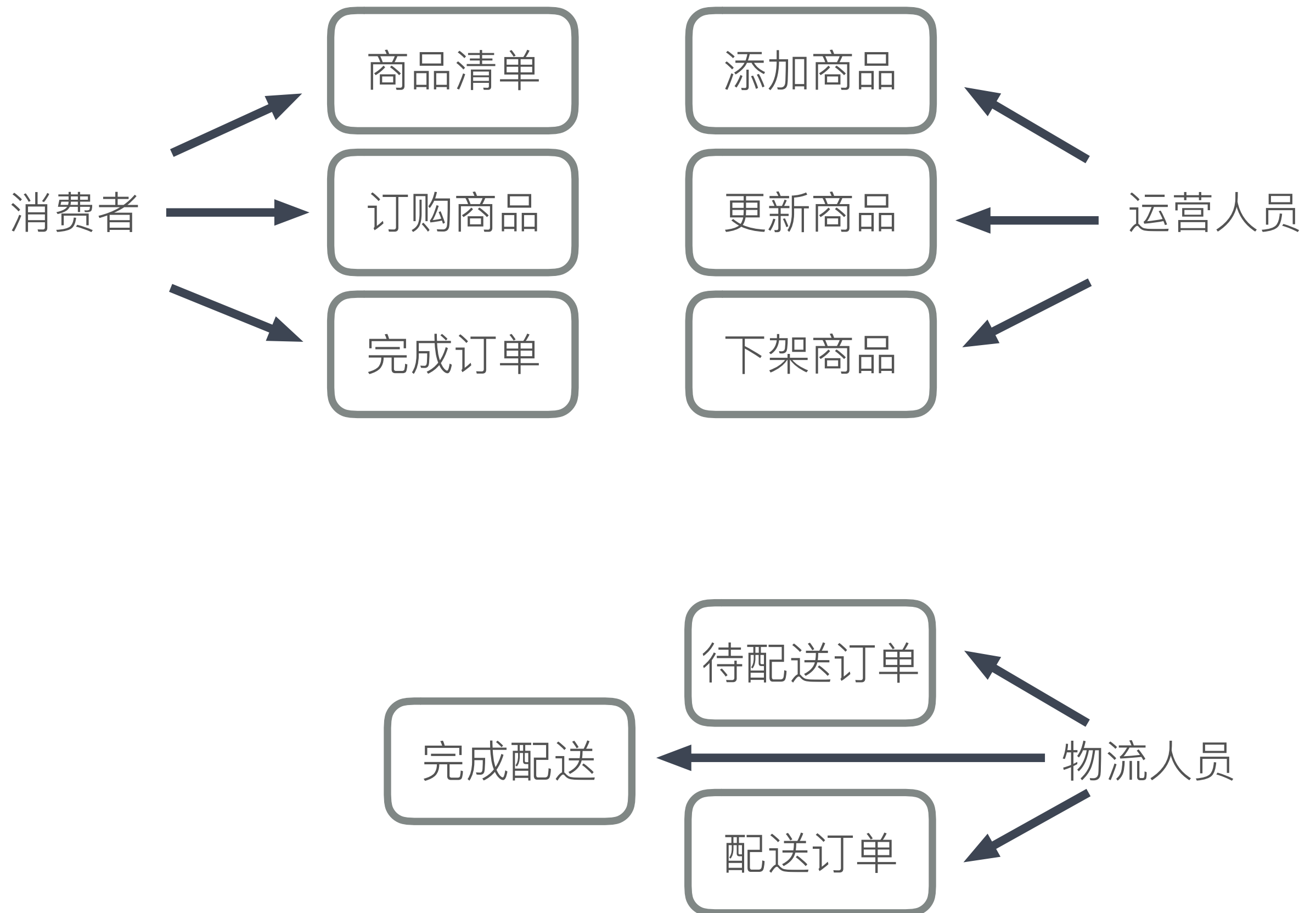
系统

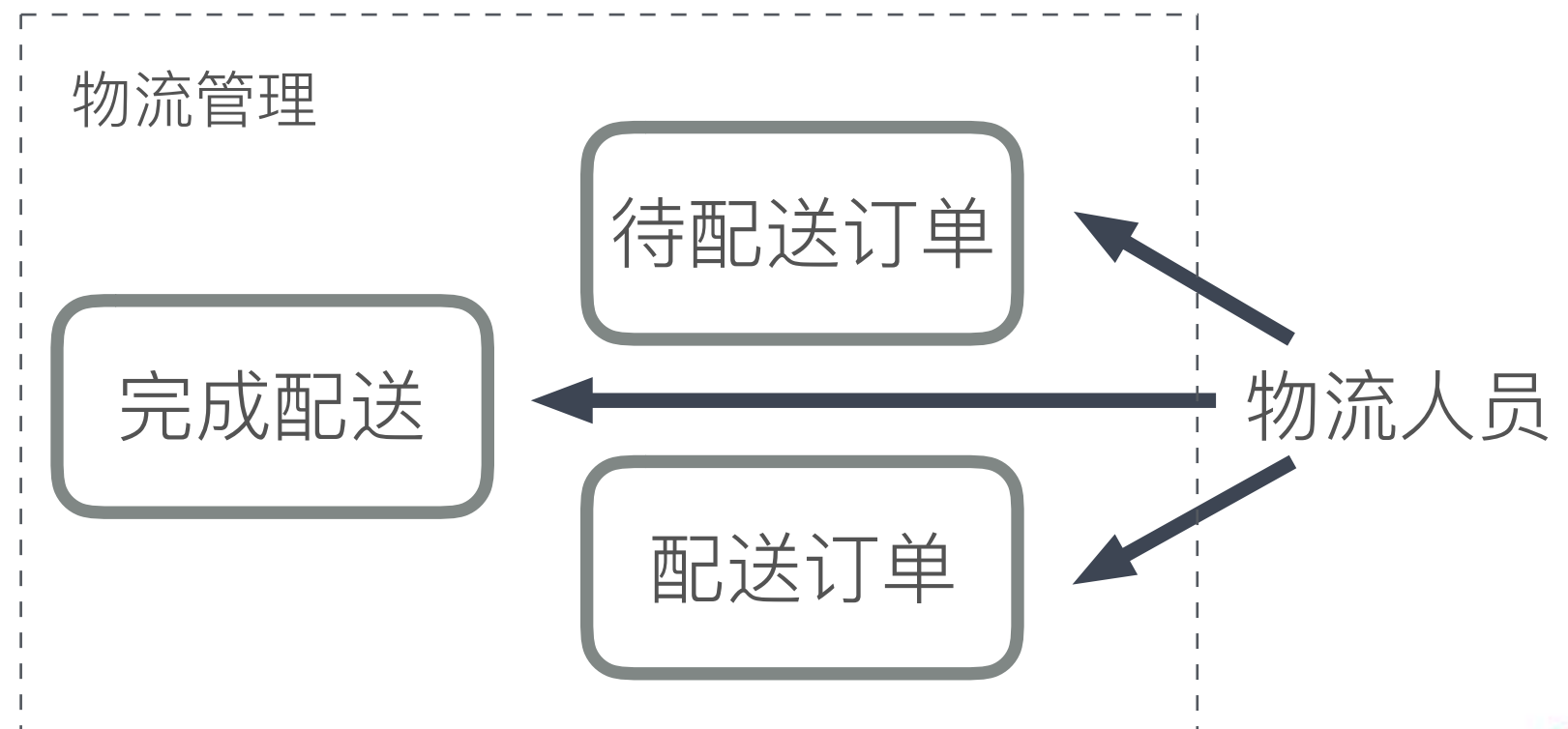
系统

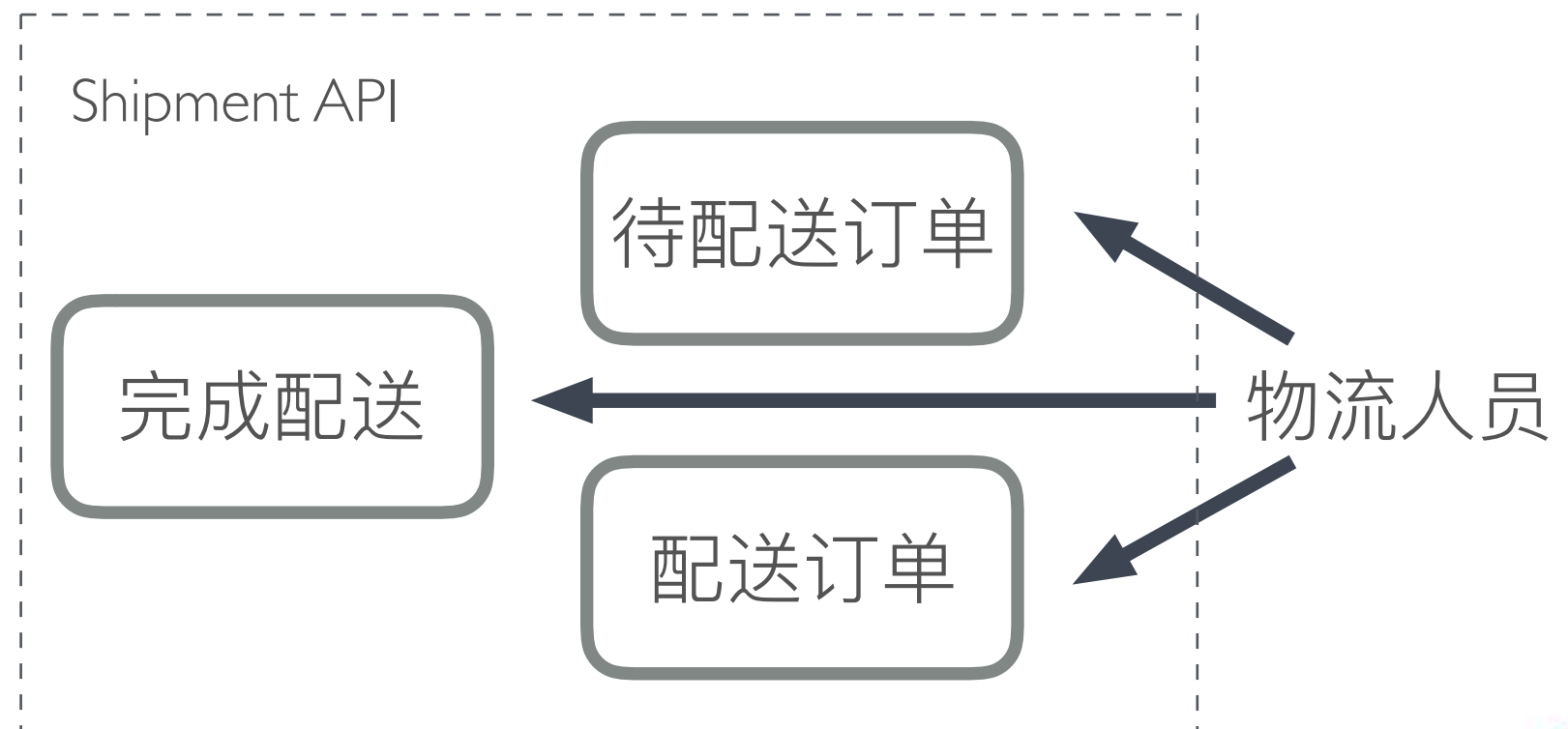
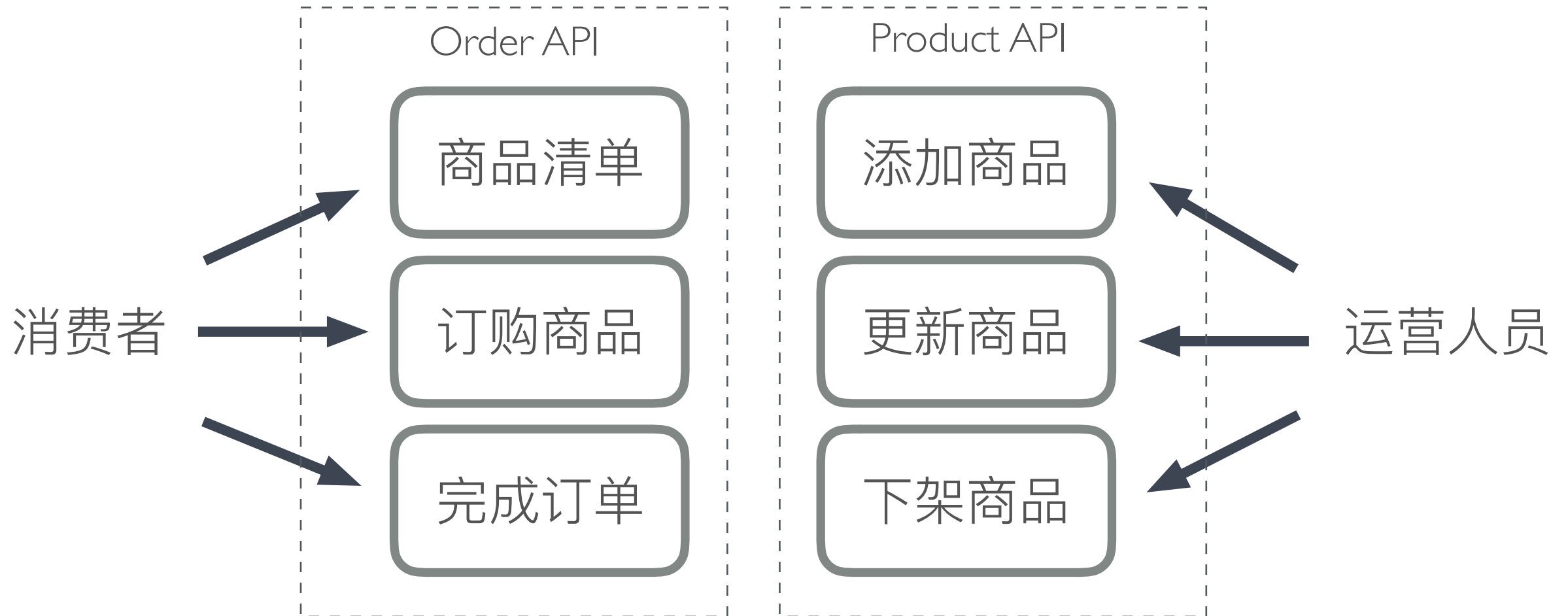
系统



# 示例项目eMail API







1

2

消费者

Order API

商品清单

订购商品

完成订单

Product API

添加商品

更新商品

下架商品

运营人员

Shipment API

待配送订单

完成配送

配送订单

物流人员

**使用领域驱动设计来找寻业务的实体、关系、状态变更和事件。**

Order API

商品清单

订购商品

完成订单

Product API

添加商品

更新商品

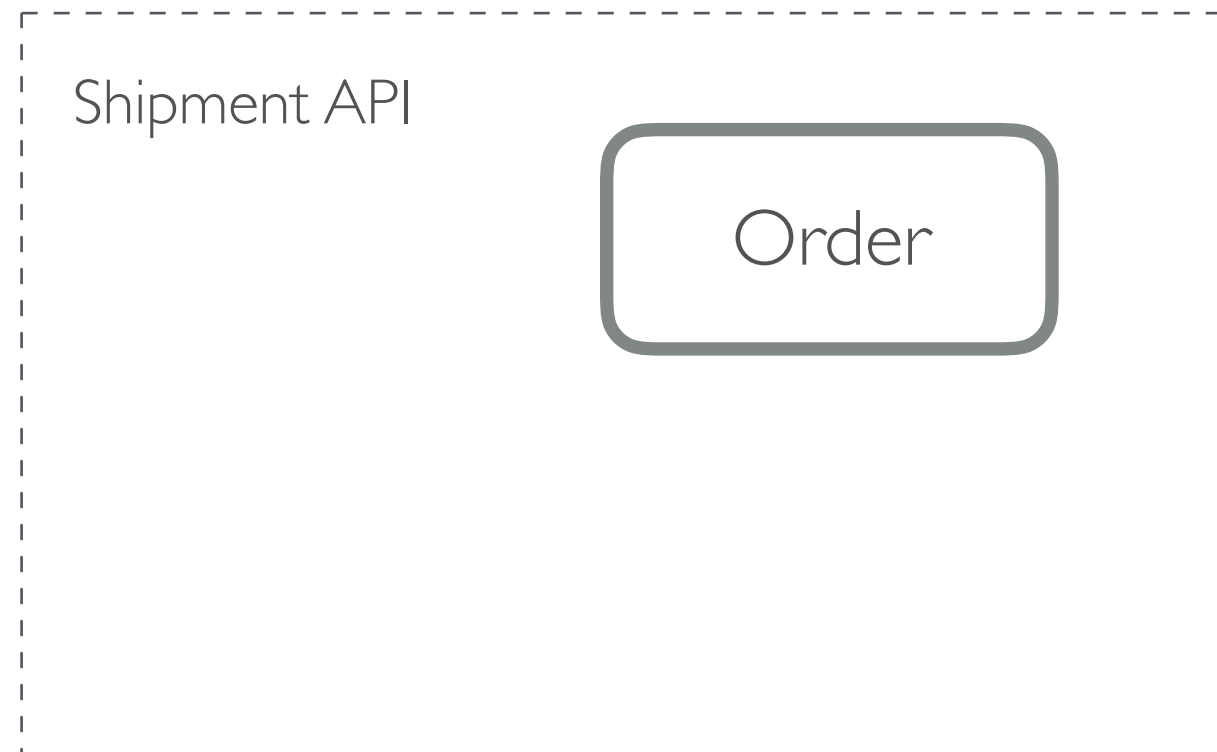
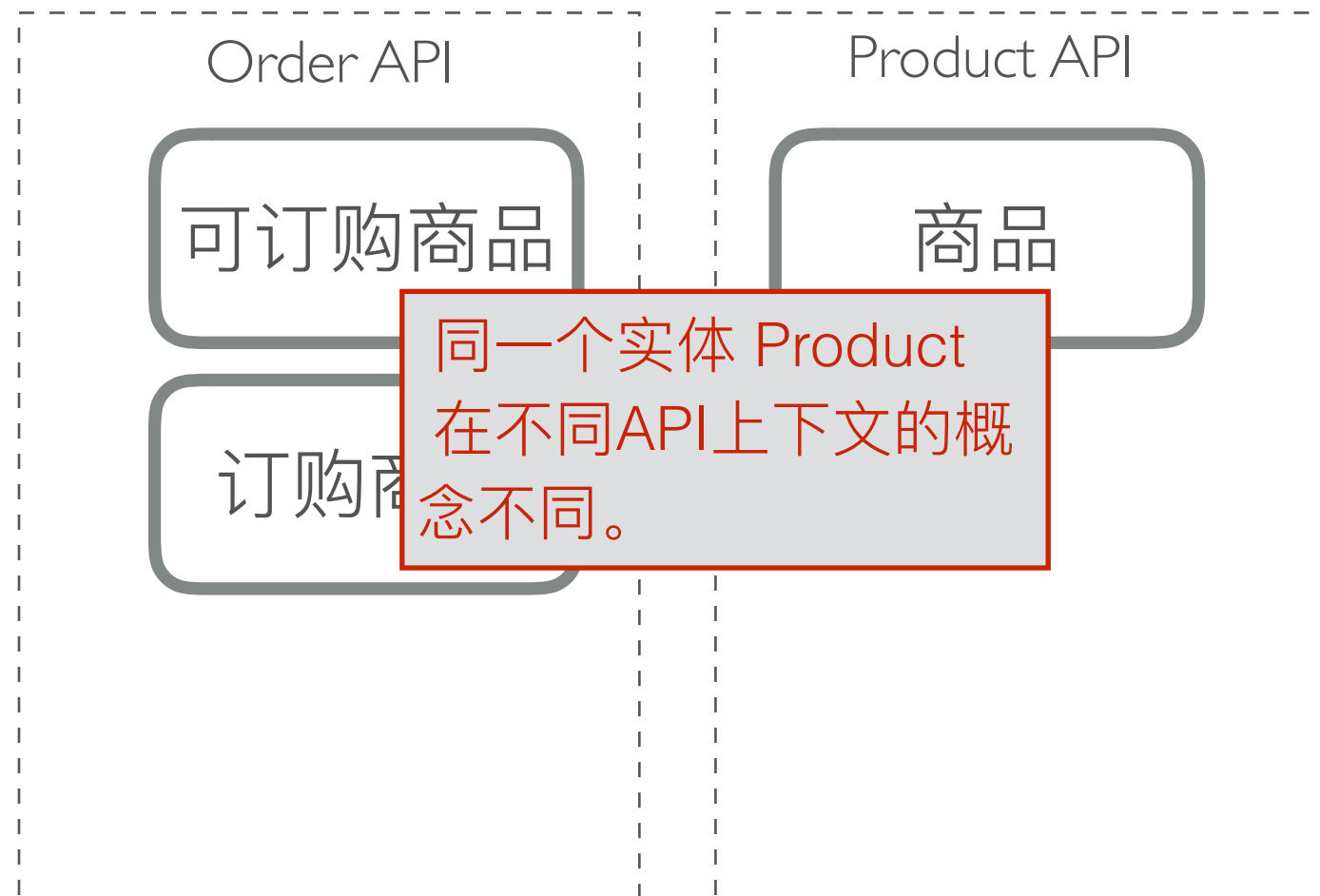
下架商品

Shipment API

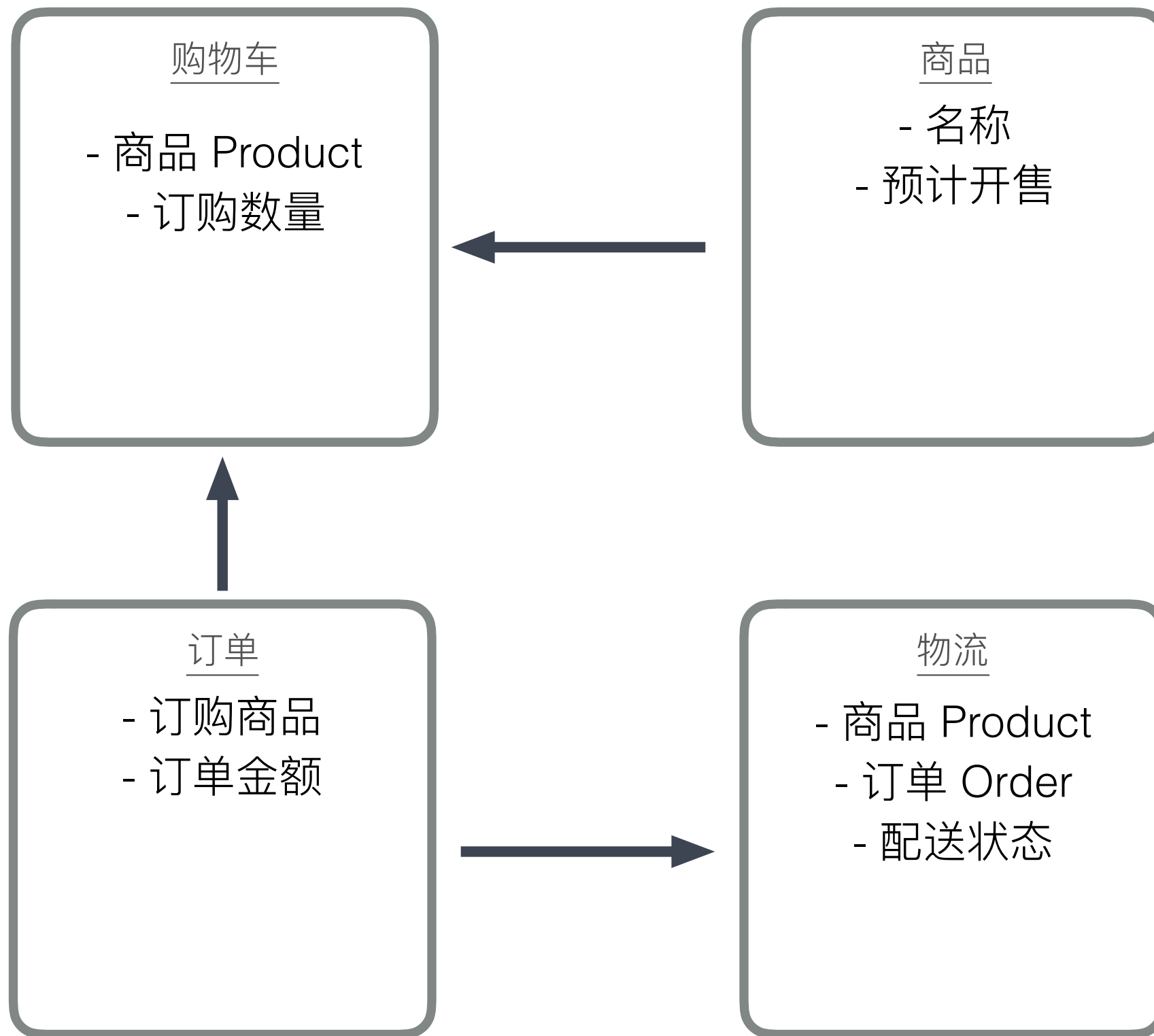
待配送订单

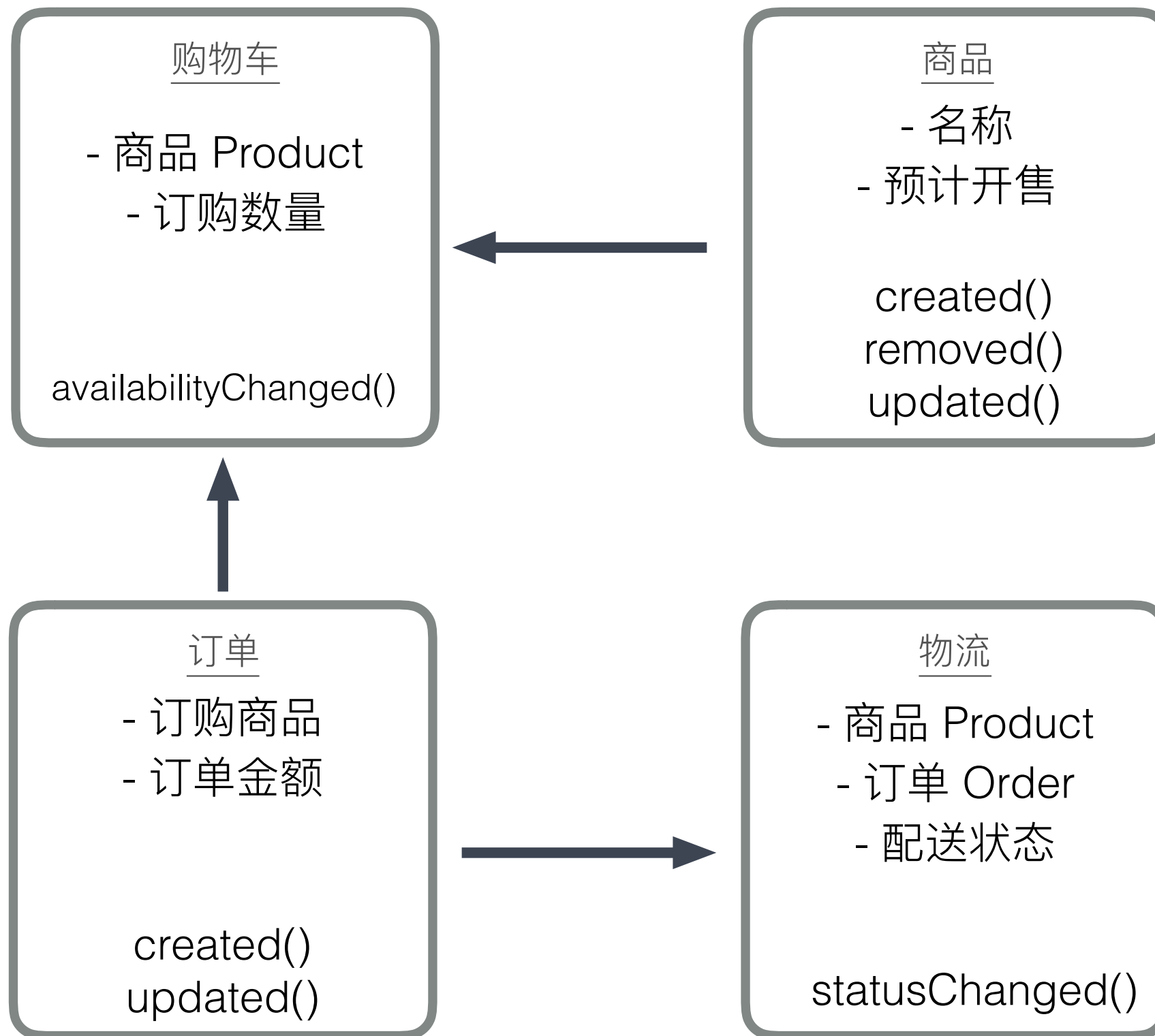
完成配送

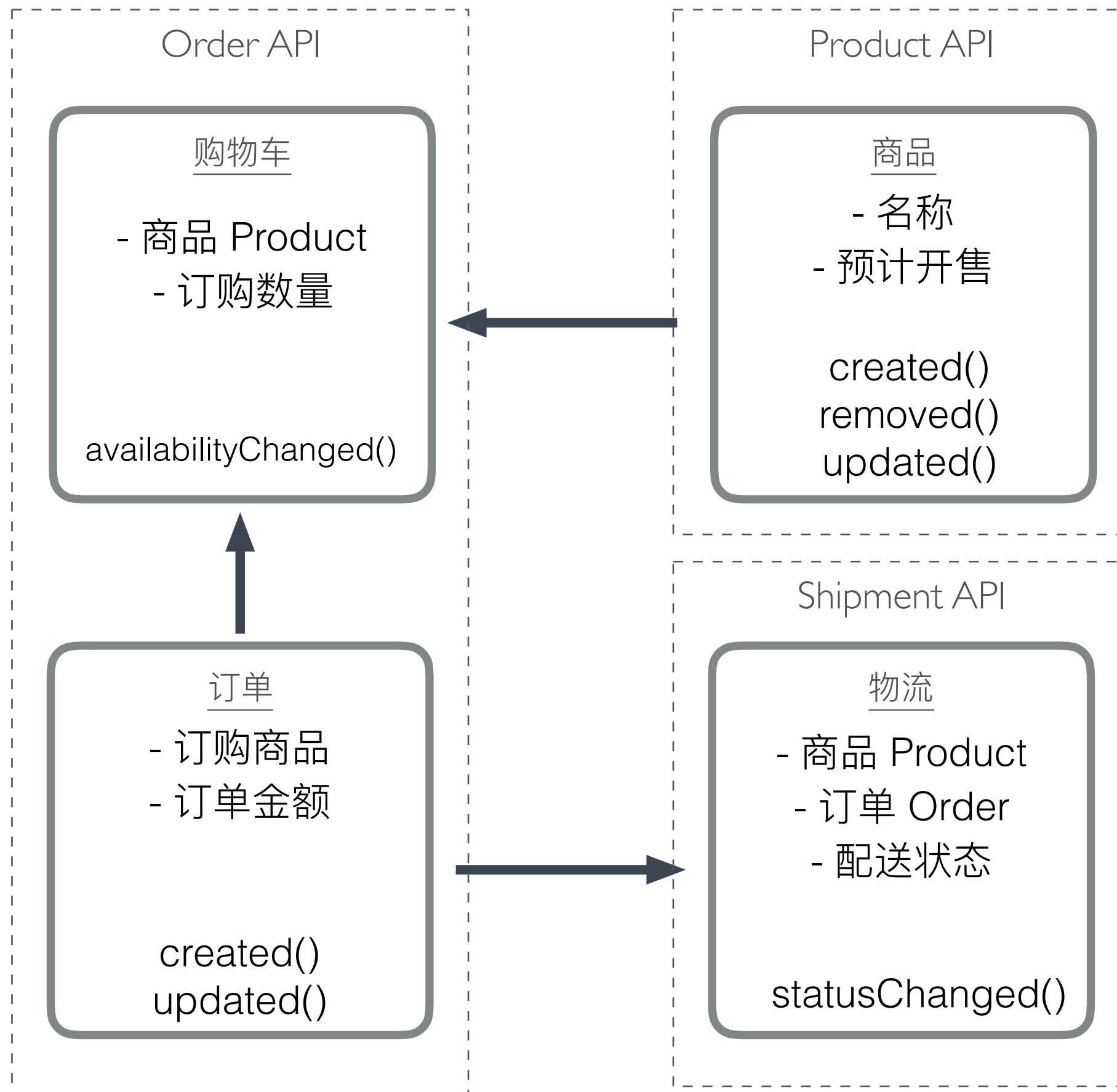
配送订单



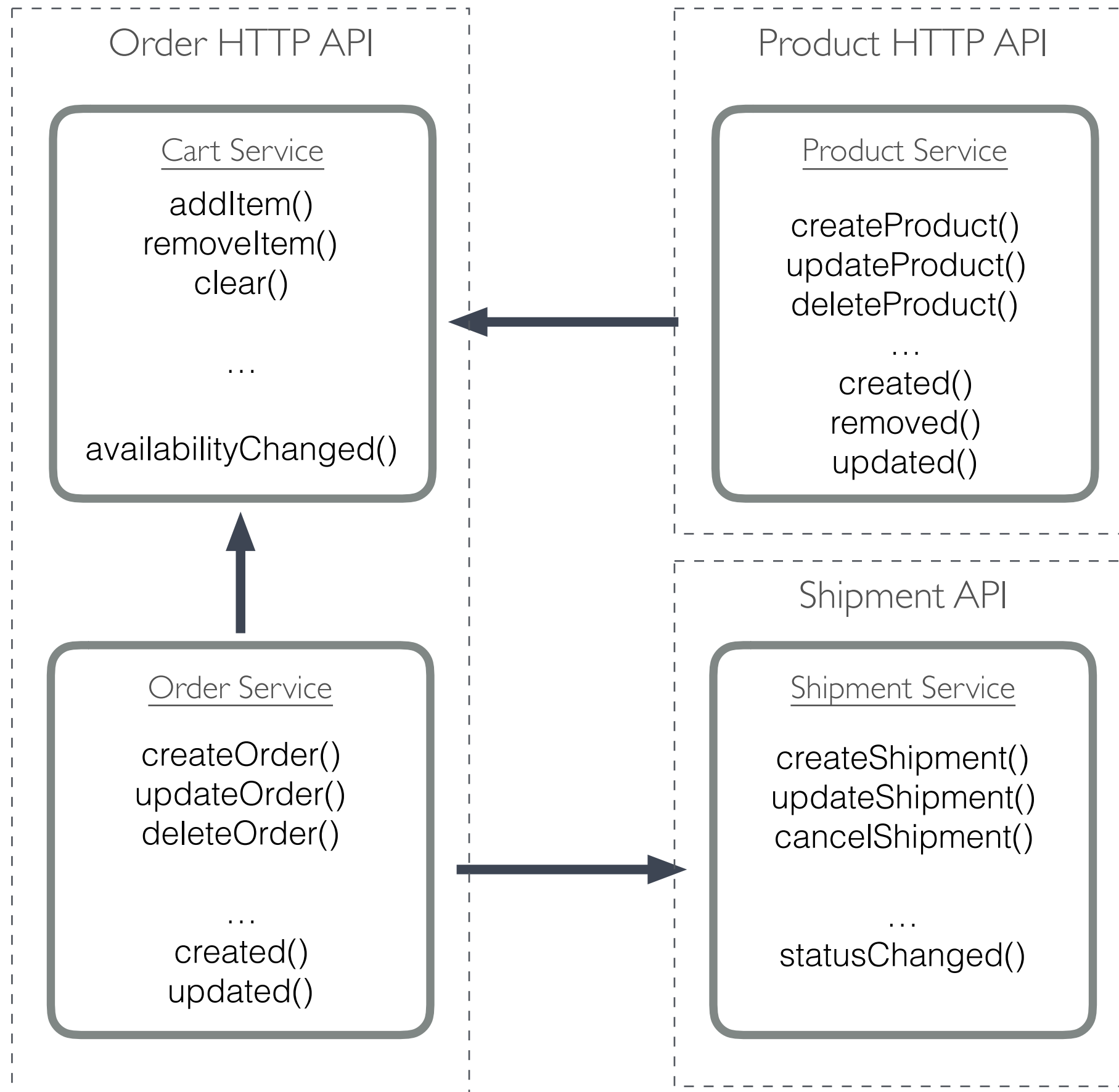








API Endpoint	描述	状态码
GET /cart	购物车详情，物品清单、数量及总金额	200 OK
POST /cart/items	向购物车添加商品，支持多件	201 Created 401 Unauthorized 400 Bad Request
PUT /cart/items/{itemId}	更新购物车里面指定商品的数量	200 OK 401 Unauthorized 404 Not Found
DELETE /cart/items	清空购物车里面的所有商品	204 No Content 401 Unauthorized
DELETE /cart/items/{itemId}	从购物车清除某指定商品	204 No Content 401 Unauthorized 404 Not Found



# 总结回顾

- 良好的API设计可帮助产品实现和表达产品远景
- API设计是粘合产品设计、软件架构和业务需求的工具
- 领域驱动设计有些概念能够帮助进行API设计
- 微服务拆分设计在API设计后进行
- 使用API来表达业务使用场景很有必要



# THANKS!

---



— 扫码了解更多 —