微服务实战篇

极客时间

扫码试看/订阅

《.NET Core 开发实战》视频课程

2.1 工程结构概览：定义应用分层及依赖关系

分层

- 领域模型层
- 基础设施层
- 应用层
- 共享层

总结

- 领域模型专注业务的设计，不依赖仓储等基础设施层
- 基础设施的仓储层仅负责领域模型的取出和存储
- 使用 CQRS 模式设计应用层
- Web API 是面向前端的交互的接口，避免依赖领域模型
- 将共享代码设计为共享包，使用私有 NuGet 仓库分发管理

2.2 定义 Entity：区分领域模型的内在逻辑和外在行为

要点

- 将领域模型字段的修改设置为私有
- 使用构造函数表示对象的创建
- 使用具有业务含义的动作来操作模型字段
- 领域模型负责对自己数据的处理
- 领域服务或命令处理者负责调用领域模型业务动作

2.3 工作单元模式（UnitOfWork）：管理好你的事务

特性

- 使用同一上下文
- 跟踪实体的状态
- 保障事务一致性

2.4 定义仓储：使用 EF Core 实现仓储层

2.5 领域事件：提升业务内聚，实现模块解耦

总结

- 由领域模型内部创建事件
- 由专有的领域事件处理类处理领域事件
- 根据实际情况来决定是否在同一事务中处理（如一致性、性能等因素）

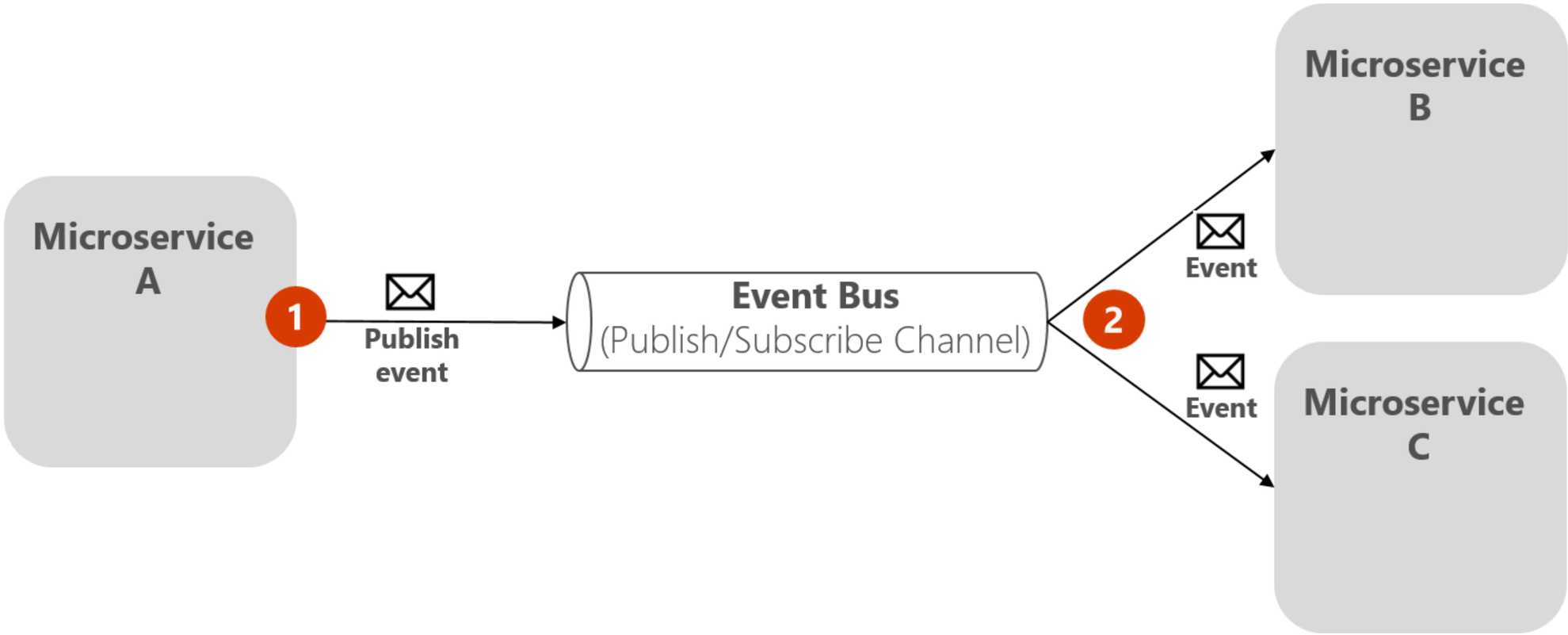
2.6 APIController：定义 API 的最佳实践

总结

- 负责用户的输入输出定义
- 负责身份认证与授权
- 与领域服务职责区分开，不承载业务逻辑

2.7 集成事件：解决跨微服务的最终一致性

集成事件工作原理


Microservice A
1
Publish event
Event Bus
(Publish/Subscribe Channel)
2
Event
Event
Microservice B
Microservice C

总结

- 集成事件是跨服务的领域事件
- 集成事件一般由领域事件驱动触发
- 不通过事务来处理集成事件（实现最终一致性）
- 仅在必要的情况下定义和使用集成事件

2.8 集成事件：使用 RabbitMQ 来实现 EventBus

实现原理

- 事件表
- 事务控制

2.9 MediatR：轻松实现命令查询职责分离模式（CQRS）

核心对象

- IMediator
- IRequest、 IRequest<T>
- IRequestHandler<in TRequest, TResponse>

2.10 MediatR：让领域事件处理更加优雅

核心对象

- IMediator
- INotification
- INotificationHandler<in TNotification>

极客时间

扫码试看/订阅

《.NET Core 开发实战》视频课程