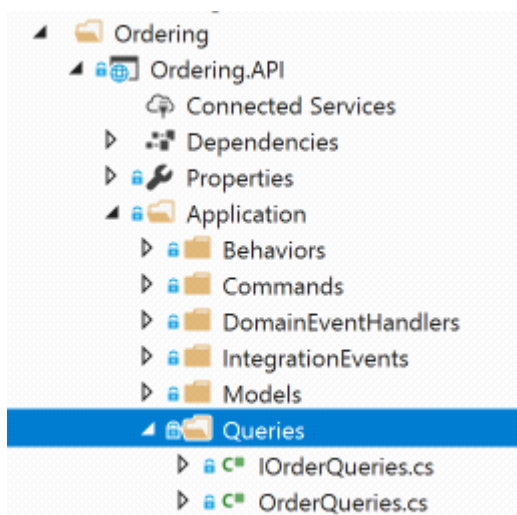
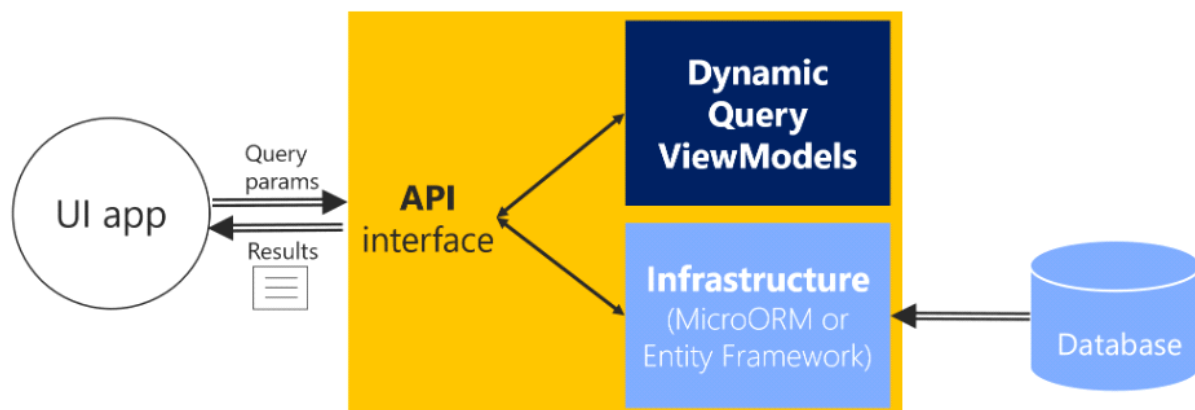


第29期-实现CQRS模式与DDD应用层

2020年5月9日 9:38

在 CQRS 微服务中实现读取查询



专为客户端应用构建的 ViewModel（不受域模型的约束）

由于执行查询是为获取客户端应用程序所需的数据，因此返回类型应根据查询所返回的数据专为客户端构建。这些模型或数据传输对象 (DTO) 称为 ViewModel。

返回数据 (ViewModel) 可以是来自数据库中多个实体或表的联接数据结果，或者是事务区域域模型中定义的一个或多个聚合中的联接数据结果。在这种情况下，因为正在创建独立于域模型的查询，所以完全忽略了聚合边界和约束，因此可随意查询可能需要的任何表和列。这种方法极大地提高了开发人员创建或更新查询的效率和灵活性。

使用 Dapper 作为微型 ORM 以执行查询

ViewModel 作为动态类型

```

public class OrderQueries : IOrderQueries
{
    public async Task<IEnumerable<dynamic>> GetOrdersAsync()
    {
        using (var connection = new SqlConnection(_connectionString))
        {
            connection.Open();
            return await connection.QueryAsync<dynamic>(@"SELECT o.[Id] as ordernumber,
o.[OrderDate] as [date],os.[Name] as [status],
SUM(oi.units*oi.unitprice) as total
FROM [ordering].[Orders] o
LEFT JOIN[ordering].[orderitems] oi ON o.Id = oi.orderid
LEFT JOIN[ordering].[orderstatus] os on o.OrderStatusId = os.Id
GROUP BY o.[Id], o.[OrderDate], os.[Name]");
        }
    }
}

```

ViewModel 作为预定义的 DTO 类

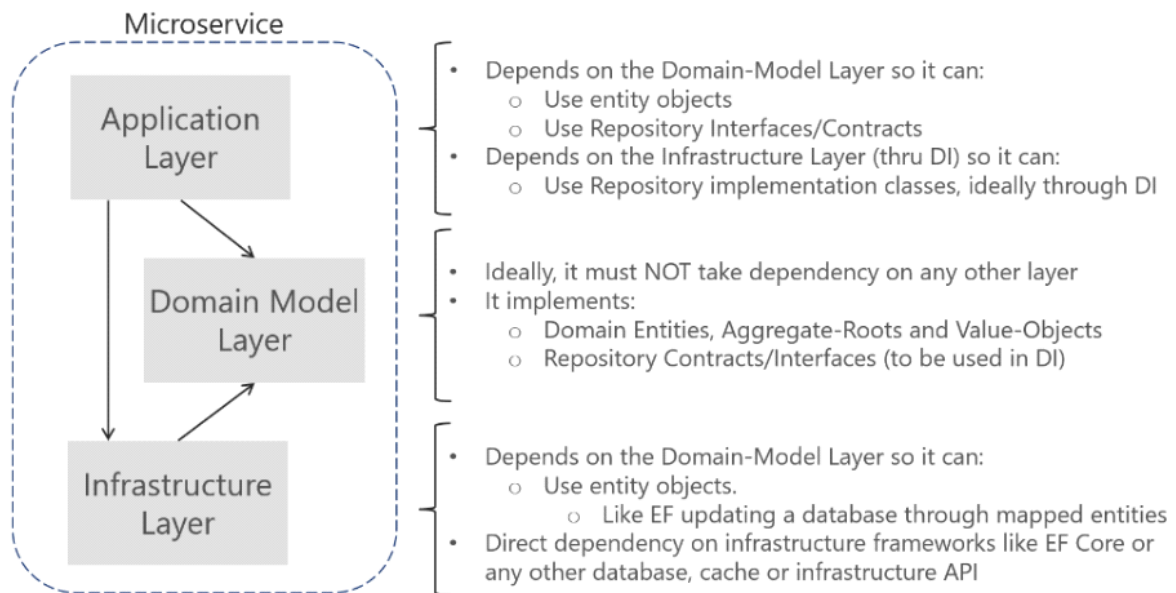
```

public class OrderQueries : IOrderQueries
{
    public async Task<IEnumerable<OrderSummary>> GetOrdersAsync()
    {
        using (var connection = new SqlConnection(_connectionString))
        {
            connection.Open();
            return await connection.QueryAsync<OrderSummary>(@"SELECT o.[Id] as ordernumber,
o.[OrderDate] as [date],os.[Name] as [status],
SUM(oi.units*oi.unitprice) as total
FROM [ordering].[Orders] o
LEFT JOIN[ordering].[orderitems] oi ON o.Id = oi.orderid
LEFT JOIN[ordering].[orderstatus] os on o.OrderStatusId = os.Id
GROUP BY o.[Id], o.[OrderDate], os.[Name]
ORDER BY o.[Id]");
        }
    }
}

```

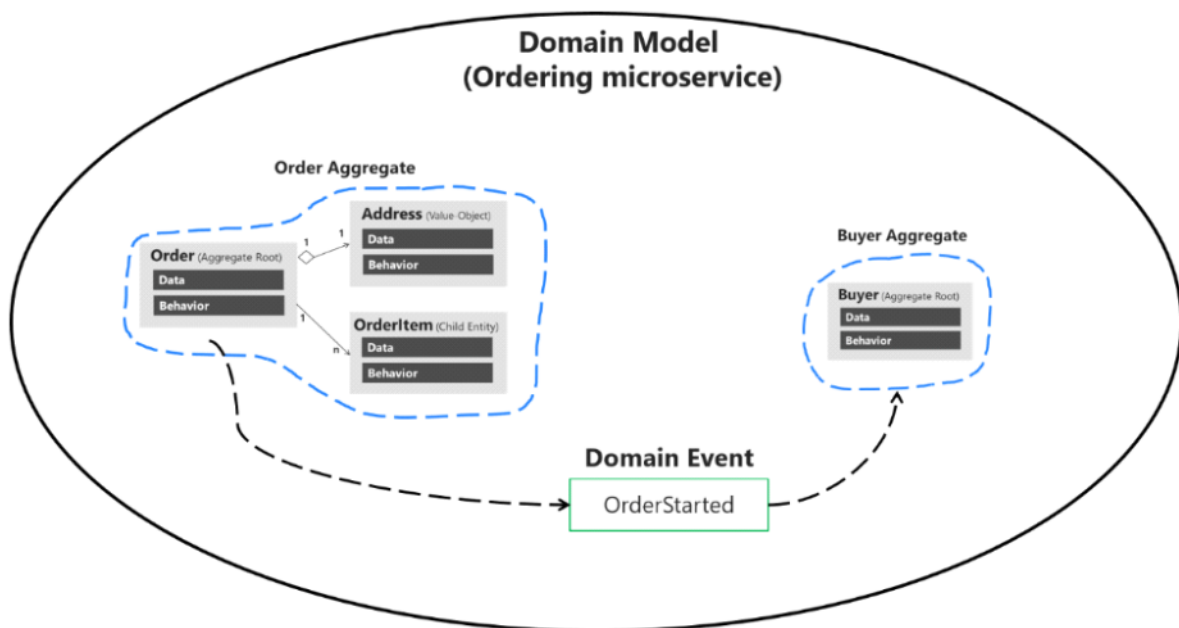
<https://github.com/StackExchange/Dapper>

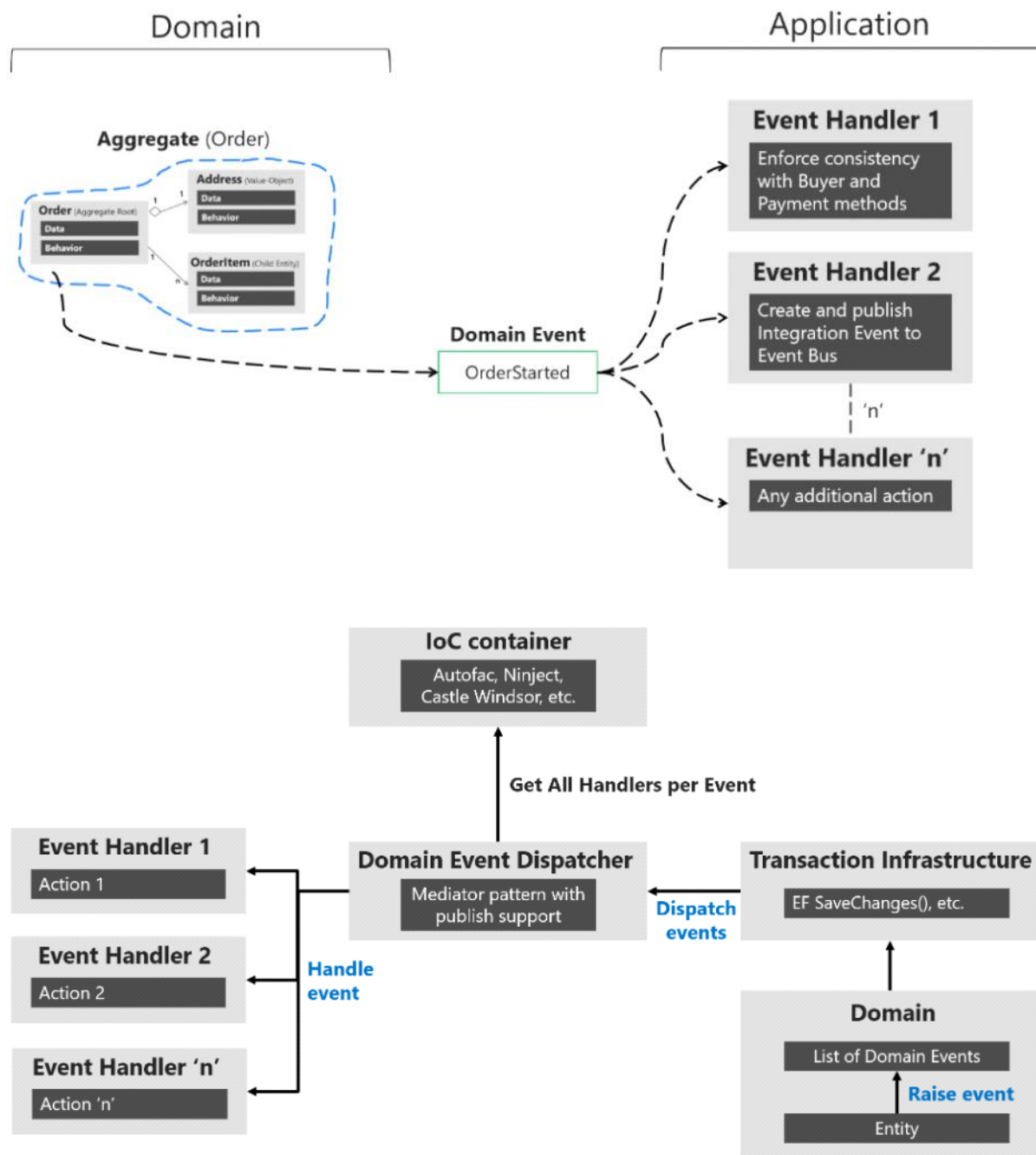
DDD 微服务中的层



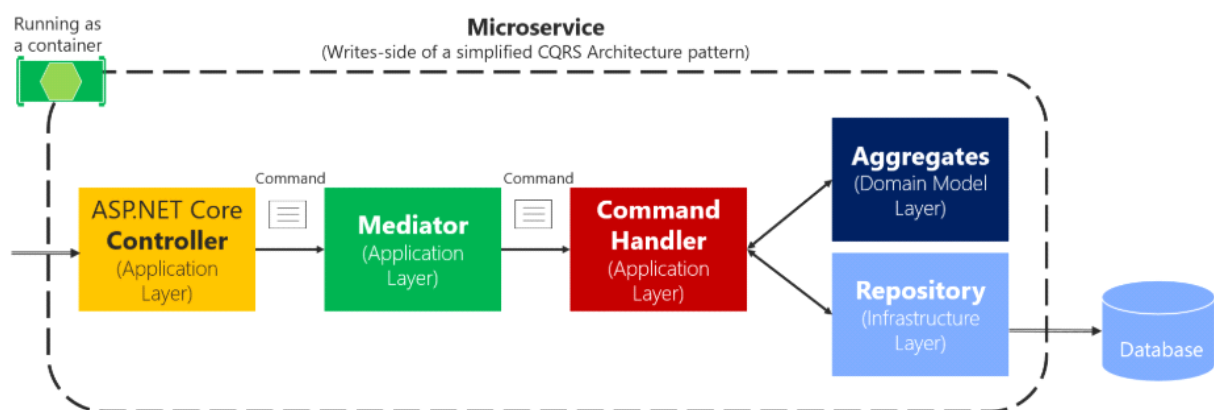
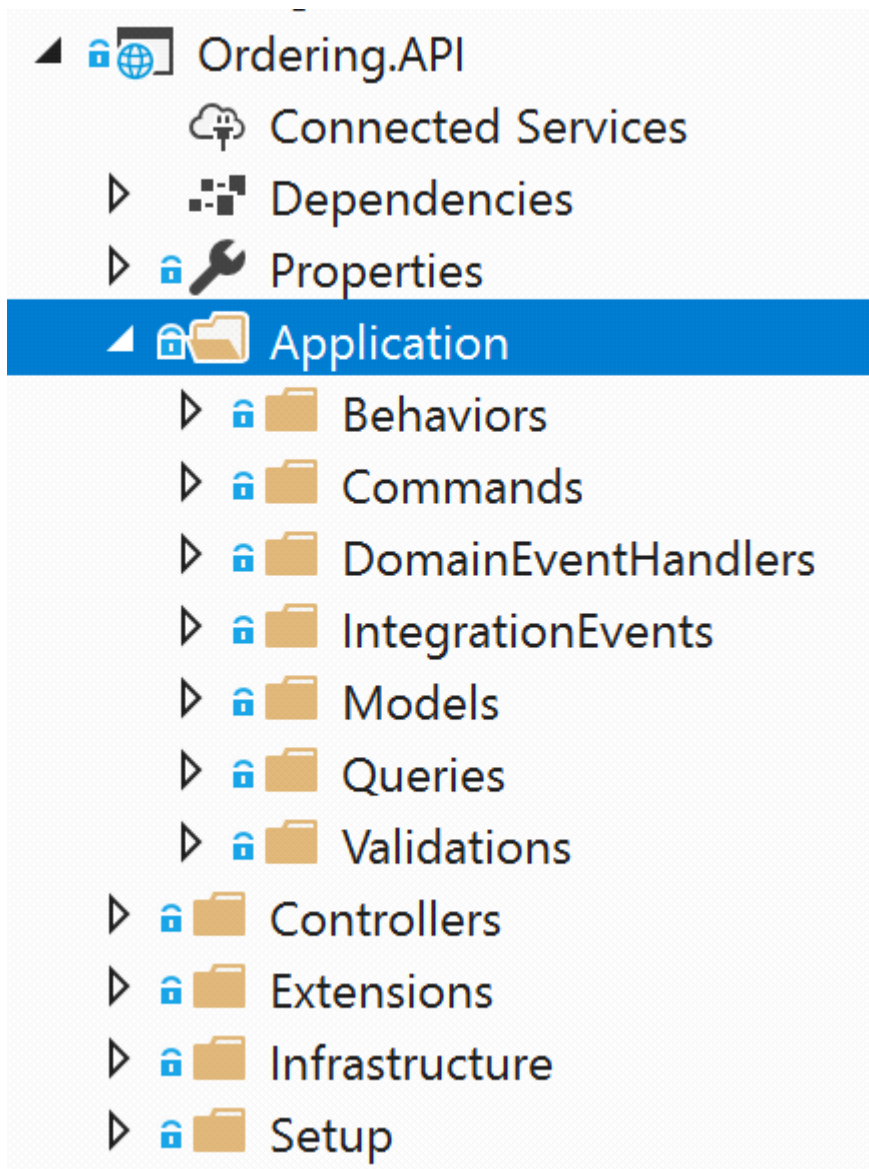
域事件的设计和实现

域事件是在相同域中跨多个聚合触发副作用的首选方式





使用 Web API 实现微服务应用层



参考资料

[使用DDD 和CQRS 模式降低微服务中的业务复杂性](#)

[Azure体系结构:命令和查询责任分离\(CQRS\) 模式](#)