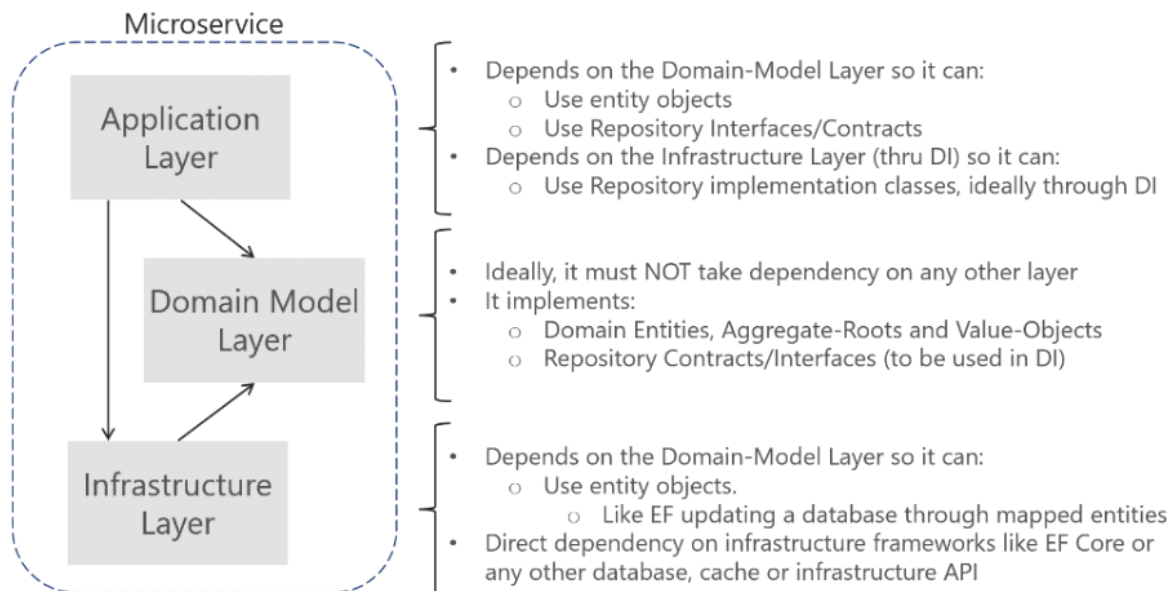


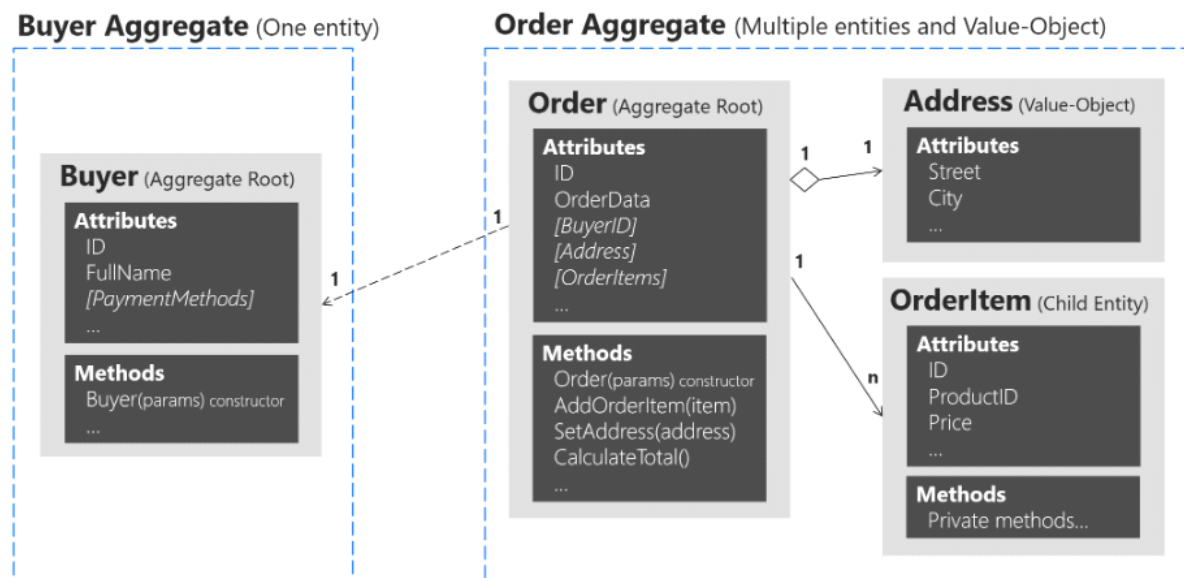
第28期-实现DDD领域层与基础设施层

2020年5月9日 9:38

DDD 微服务中的层

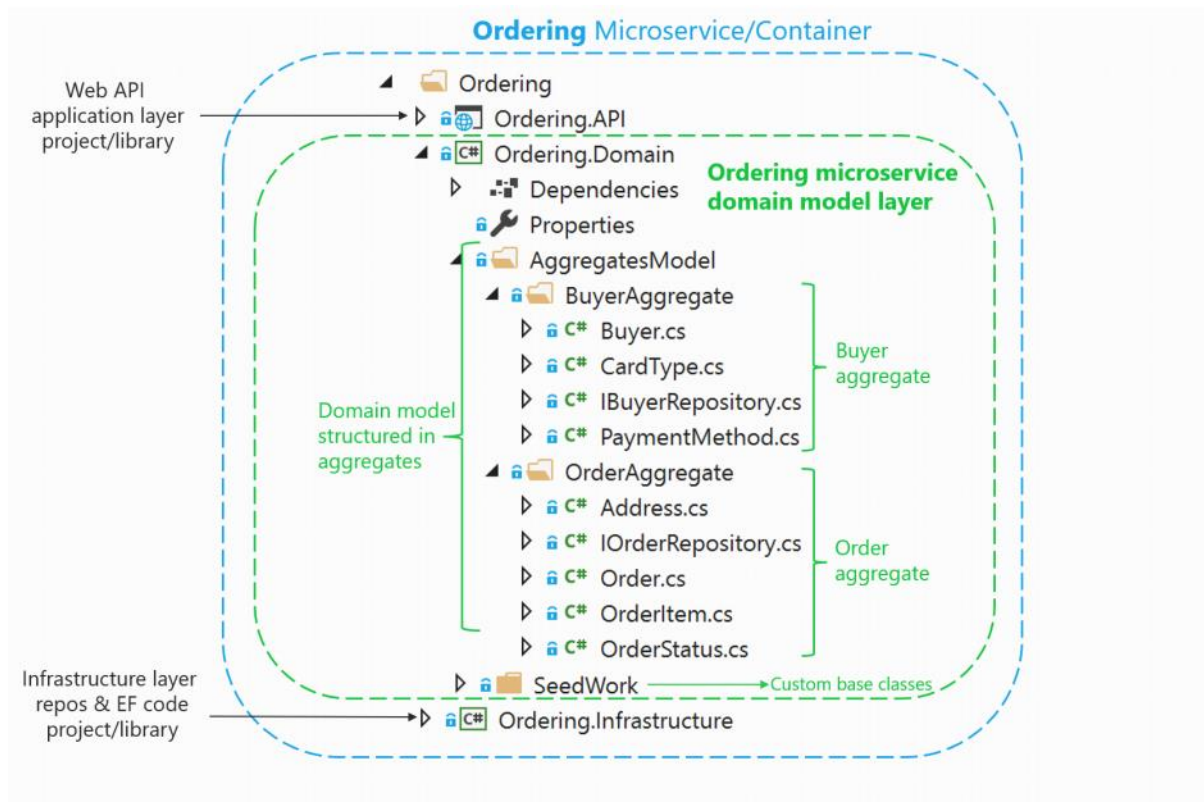


设计微服务域模型

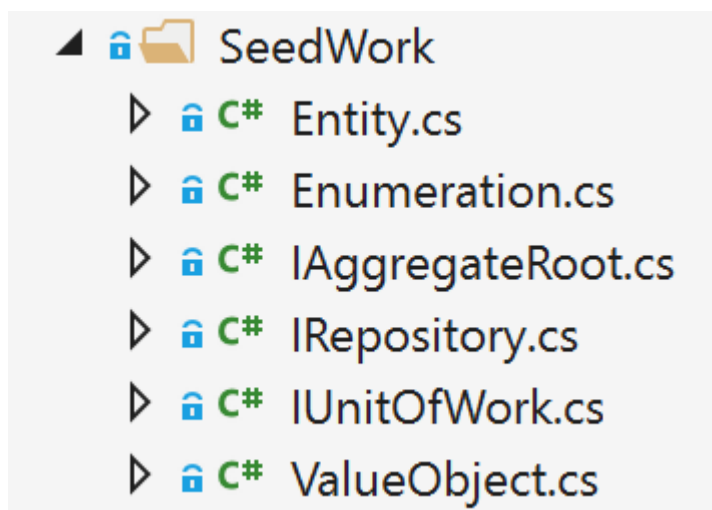


```
public class Order : Entity, IAggregateRoot
{
    private DateTime _orderDate;
    public Address Address { get; private set; }
    private int? _buyerId; //FK pointing to a different aggregate root
    public OrderStatus OrderStatus { get; private set; }
    private readonly List<OrderItem> _orderItems;
    public IReadOnlyCollection<OrderItem> OrderItems => _orderItems;
```

```
// ... Additional code
}
```



Seedwork (适用于域模型的可重用基类和接口)



使用枚举类（而不是枚举类型）

```
public abstract class Enumeration : IComparable
{
    public string Name { get; private set; }

    public int Id { get; private set; }
```

```

protected Enumeration(int id, string name)
{
    Id = id;
    Name = name;
}

public override string ToString() => Name;

public static IEnumerable<T> GetAll<T>() where T : Enumeration
{
    var fields = typeof(T).GetFields(BindingFlags.Public |
indingFlags.Static |
indingFlags.DeclaredOnly);

    return fields.Select(f => f.GetValue(null)).Cast<T>();
}

public override bool Equals(object obj)
{
    var otherValue = obj as Enumeration;

    if (otherValue == null)
        return false;

    var typeMatches = GetType().Equals(obj.GetType());
    var valueMatches = Id.Equals(otherValue.Id);

    return typeMatches && valueMatches;
}

public int CompareTo(object other) => Id.CompareTo(((Enumeration)
other).Id);

// Other utility methods ...
}

public class CardType : Enumeration
{
    public static readonly CardType Amex = new CardType(1, "Amex");
    public static readonly CardType Visa = new CardType(2, "Visa");
    public static readonly CardType MasterCard = new CardType(3, "MasterCard");

    public CardType(int id, string name)
        : base(id, name)
    {
    }
}

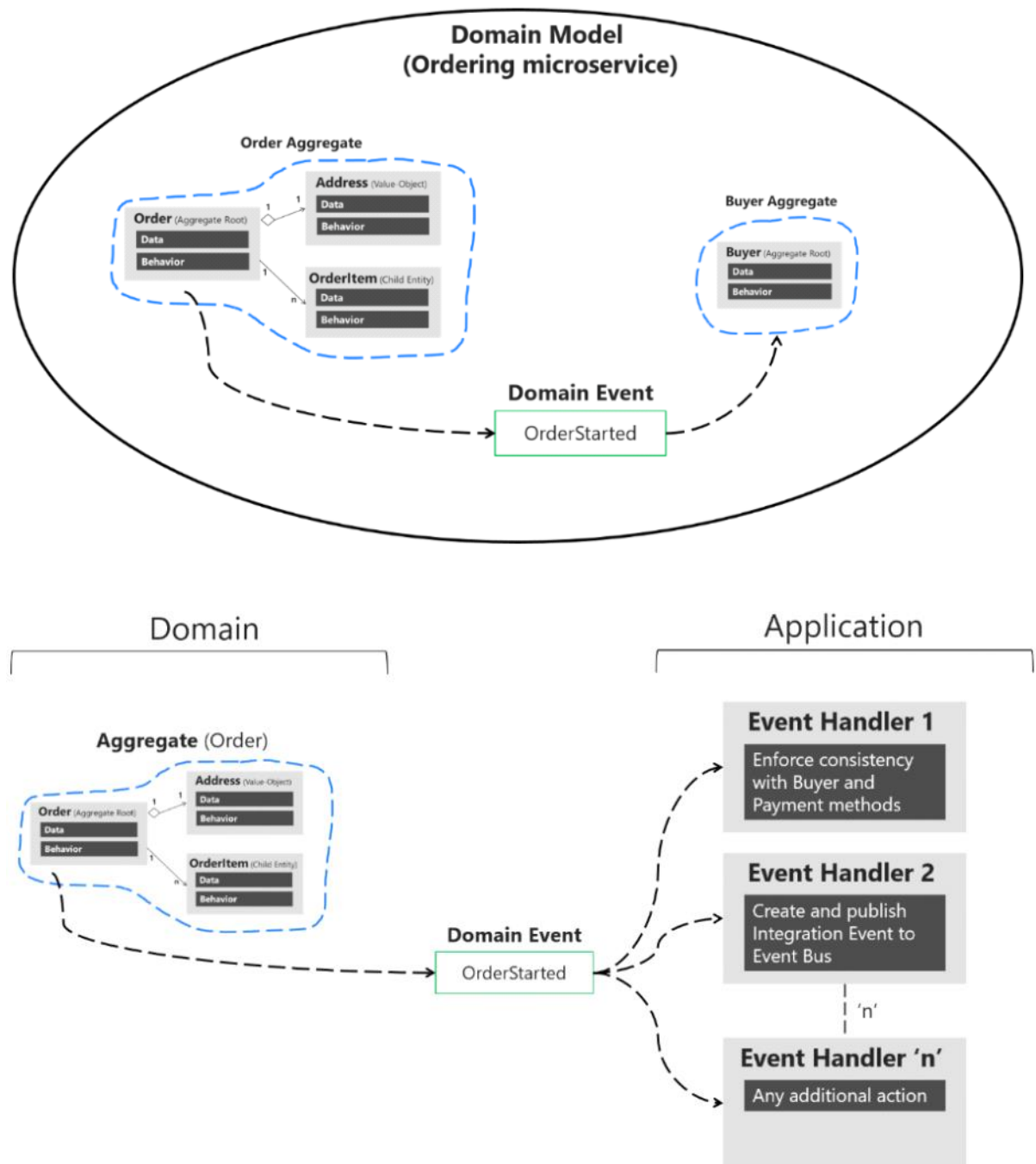
```

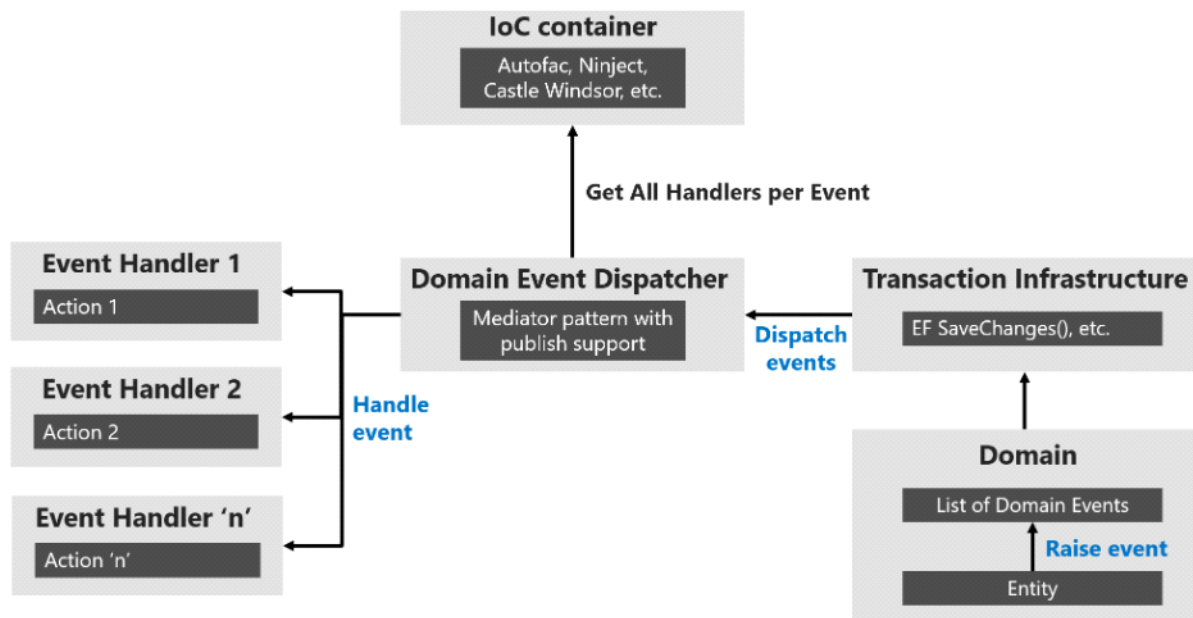
B

B

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





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