Xelseor
Consulting Group

Modular Monolith Patterns

For .Net Engineers

Building Scalable, Maintainable Applications in .Net

Johnathon Smith
Architecture Consultant

More Information
www.xelseor.com

Xelseor
Consulting Group

Modular Monolith Patterns

Agenda

1. Discuss What a Modular Monolith is and why we would use it.
2. Discuss the structure of a Modular Monolith
3. Discuss In Memory Communication with MediatR
4. Discuss the Saga Pattern

We will use .Net Aspire to orchestrate our local environment, but we will not cover how to utilize Aspire in the lecture.

More Information
www.xelseor.com

Xelseor
Consulting Group

Modular Monolith Patterns

Get the Code!

Repo URL
https://github.com/JohnnyDevCraft/ShipSim

What is a Modular Monolith, And Why Use It?

Xelseor Consulting Group

A **Modular Monolith** is a single-deployable application where functionality is divided into well-structured, independent modules that communicate internally while remaining loosely coupled.

✅ Key Characteristics:

- 📦 Encapsulated Modules
- 🔄 Internal Communication
- 📂 Independent Data Handling
- 🚀 Scalability without Complexity

Compared to Other Architectures...

Xelseor Consulting Group

Monolith

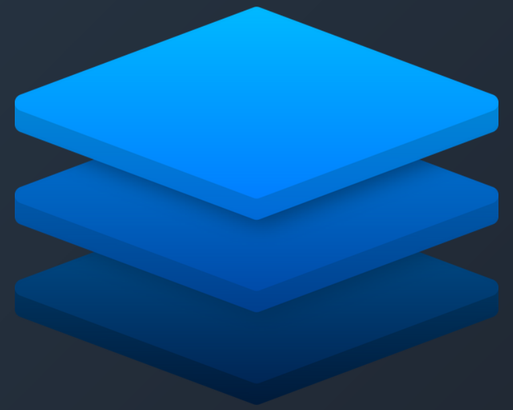
Modular Monolith

Micro-Services

Internal Architectures for
Modular Monoliths..

Xelseor
Consulting Group

Integrated



Layered

Demo: Application Structure StarShip Simulator

What is Tight Coupling in, an N-Tiered Monolith?

Xelseor Consulting Group

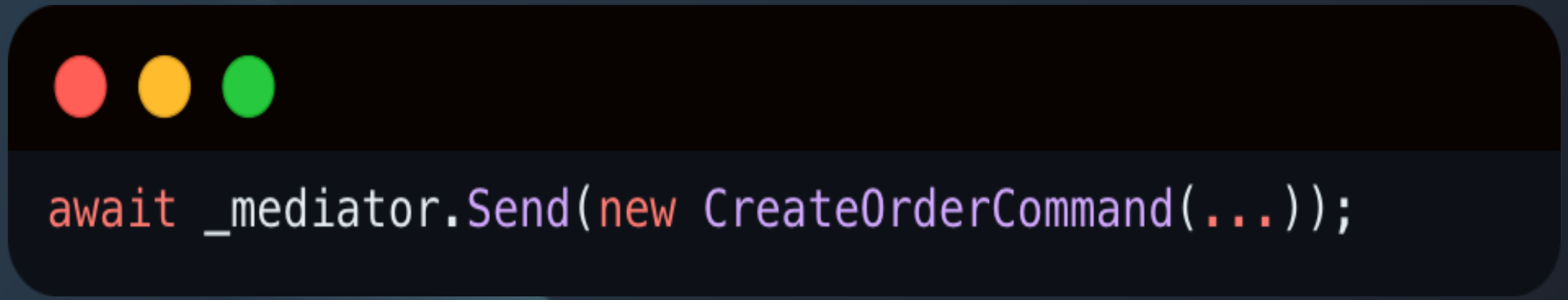
```
public class OrderService
{
    private readonly ProductService _productService; ❌ // Direct dependency

    public OrderService(ProductService productService)
    {
        _productService = productService;
    }
}
```

- Tightly Coupled to Products
- Easy to Break other Systems When Upgrading
- Slower to Change Things because of interdependencies

The Mediator Pattern
Breaks Tight Coupling

Xelseor
Consulting Group

- Removes Direct Dependencies on Services
- Allows for implementations to be easily swapped out.
- Allows us to encapsulate our Domain inside our Module.



The Mediator Pattern
Commands, Queries, and Requests

Xelseor Consulting Group

```
public record CreateOrderCommand(int ProductId, int Quantity) : IRequest<int>;
```

```
public class CreateOrderCommandHandler : IRequestHandler<CreateOrderCommand, int>
{
    private readonly IOrderRepository _orderRepository;

    public CreateOrderCommandHandler(IOrderRepository orderRepository)
    {
        _orderRepository = orderRepository;
    }

    public async Task<int> Handler(CreateOrderCommand request, CancellationToken cancellationToken)
    {
        var orderId = await _orderRepository.CreateAsync(request.ProductId, request.Quantity);
        return orderId;
    }
```

The Mediator Pattern

Commands, Queries, and Requests

Xelseor
Consulting Group

```
[HttpPost]
public async Task<IActionResult> CreateOrder([FromBody] CreateOrderRequest request)
{
    var orderId = await _mediator.Send(new CreateOrderCommand(request.ProductId, request.Quantity));
    return Ok(orderId);
}
```

The Mediator Pattern
Notifications & Events

Xelseor
Consulting Group

```
public record OrderCreatedEvent(int OrderId) : INotification;
```

```
public class SendEmailOnOrderCreatedHandler : INotificationHandler<OrderCreatedEvent>
{
    private readonly IEmailService _emailService;

    public SendEmailOnOrderCreatedHandler(IEmailService emailService)
    {
        _emailService = emailService;
    }

    public async Task Handle(OrderCreatedEvent notification, CancellationToken cancellationToken)
    {
        await _emailService.SendOrderConfirmation(notification.OrderId);
    }
}
```

Demo: MediatR Requests and Handlers : How it works now.

New Feature: We need to allow users to update thier First and Last Names. We need to ensure that when they do cached users in the ShipModule are updated.

New Feature

Create ChangeNamesForUserRequest / Result


```
using MediatR;
using ShipSim.Players.Module.Contracts.ViewModels;

namespace ShipSim.Players.Module.Contracts.Requests;

public record ChangeNamesForUserRequest(string FirstName, string LastName) : IRequest<ChangeNamesForUserRequestResult>;

public record ChangeNamesForUserRequestResult(PlayerDto PlayerDto);
```

New Feature

Create ChangeNamesForUserCommand / Result

```
using MediatR;
using ShipSim.Players.Module.Contracts.ViewModels;

namespace ShipSim.Players.Module.Commands;

public record ChangeNamesForUserCommand(string FirstName, string LastName, string Email) :
IRequest<ChangeNamesForUserCommandResult>;

public record ChangeNamesForUserCommandResult(PlayerDto Player);
```

New Feature

Create ChangeNamesForUserRequestHandler

Xelseor Consulting Group

```
using System.Security.Claims;
using MediatR;
using Microsoft.AspNetCore.Http;
using Microsoft.Extensions.Logging;
using ShipSim.Core.Exceptions;
using ShipSim.Players.Module.Commands;
using ShipSim.Players.Module.Contracts.Requests;
using ShipSim.Players.Module.Queries;

namespace ShipSim.Players.Module.Handlers.Request;

public class ChangeNamesForUserRequestHandler(IMediator mediator, ILogger<ChangeNamesForUserRequestHandler> logger, IHttpContextAccessor accessor) : IRequestHandler<ChangeNamesForUserRequest,
ChangeNamesForUserRequestResult>
{
    public async Task<ChangeNamesForUserRequestResult> Handle(ChangeNamesForUserRequest request, CancellationToken cancellationToken)
    {
        var email = accessor.HttpContext?.User.Claims.FirstOrDefault(c => c.Type == ClaimTypes.Email)?.Value;

        if (email == null)
        {
            var ex =  new UserUnauthenticatedException();
            logger.LogError(ex, "User is not authenticated");
            throw ex;
        }

        logger.LogInformation("Change names for user: {Email}", email);

        var user = await mediator.Send(new GetUserByEmailQuery(email), cancellationToken);

        var updatedUser = await mediator.Send(new ChangeNamesForUserCommand(request.FirstName, request.LastName, email), cancellationToken);

        return new ChangeNamesForUserRequestResult(updatedUser.Player);
    }
}
```

New Feature

Create ChangeNamesForUserCommandHandler

Xelseor Consulting Group

```
using AutoMapper;
using MediatR;
using Microsoft.EntityFrameworkCore;
using ShipSim.Players.Module.Commands;
using ShipSim.Players.Module.Contracts.Events;
using ShipSim.Players.Module.Contracts.Exceptions;
using ShipSim.Players.Module.Contracts.ViewModels;
using ShipSim.Players.Module.DataAccess;

namespace ShipSim.Players.Module.Handlers.Command;

internal class ChangeNamesForUserCommandHandler(PlayersDbContext db, IMapper mapper, IMediator mediator) : IRequestHandler<ChangeNamesForUserCommand, ChangeNamesForUserCommandResult>
{
    public async Task<ChangeNamesForUserCommandResult> Handle(ChangeNamesForUserCommand request, CancellationToken cancellationToken)
    {
        // TODO: Implement the logic to change the user name
        var player = await db.Users.FirstOrDefaultAsync(x => x.Email == request.Email, cancellationToken);

        if (player == null)
        {
            throw new UserByEmailNotFoundException(request.Email);
        }

        player.FirstName = request.FirstName;
        player.LastName = request.LastName;

        await db.SaveChangesAsync(cancellationToken);
        var dto = mapper.Map<PlayerDto>(player);

        // TODO: Add the event code here

        return new ChangeNamesForUserCommandResult(mapper.Map<PlayerDto>(player));
    }
}
```

New Feature

Update Endpoints to include Patch EP

```
app.MapPut("/players/my-name", async (IMediator mediator, ChangeNamesForUserRequest request) =>
{
    var result = await mediator.Send(request);
    return Results.Ok(result);
}).RequireAuthorization().RequireCors();
```

New Feature

Test Our Work…

New Feature

Create UserUpdatedEvent

Xelseor Consulting Group

```
using ShipSim.Core;
using ShipSim.Players.Module.Contracts.ViewModels;

namespace ShipSim.Players.Module.Contracts.Events;

public record UserUpdatedEvent(DateTime EventTime, string Email, PlayerDto Player): IEvent;
```

New Feature

Update our Change Names Command

Replace:

```
// TODO: Add the event code here
```

With:

```
await mediator.Publish(new UserUpdatedEvent(DateTime.Now,dto.Email,dto), cancellationToken);
```

New Feature

Create UpdateUserCacheOnUserModifiedEventHandler

Xelseor Consulting Group

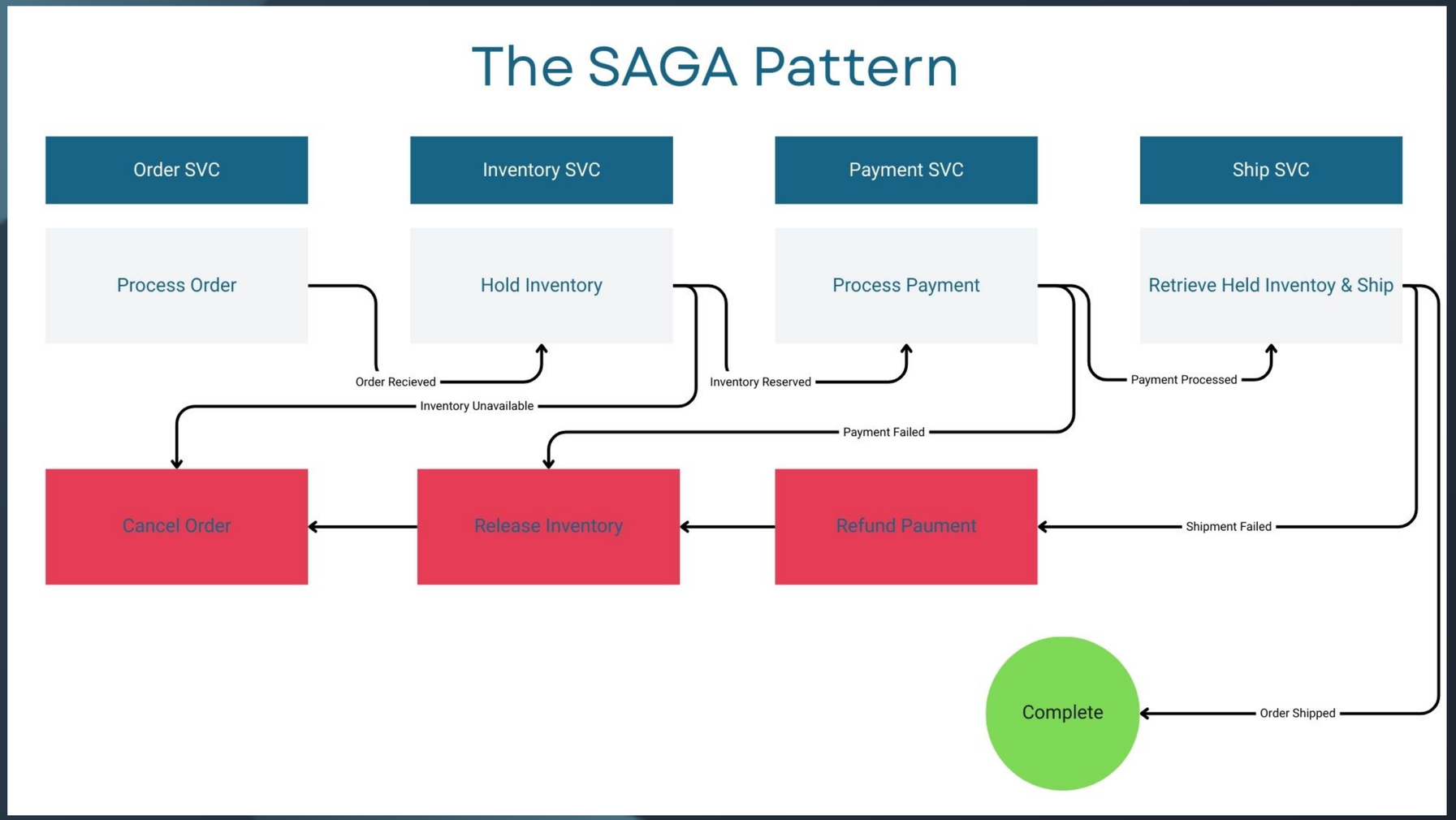
```
using MediatR;
using ShipSim.AspireConstants;
using ShipSim.Players.Module.Contracts.Events;
using ShipSim.Ship.Module.Caching;

namespace ShipSim.Ship.Module.EventHandler;

internal class UpdateUserCacheOnUserModifiedEventHandler(ICacheManager cm): INotificationHandler<UserUpdatedEvent>
{
    public async Task Handle(UserUpdatedEvent notification, CancellationToken cancellationToken)
    {
        if(await cm.ExistsAsync(string.Format(Defaults.ShipModule.RedisPlayersPrefix, notification.Player.Email)))
        {
            await cm.RemoveAsync(string.Format(Defaults.ShipModule.RedisPlayersPrefix, notification.Player.Email));
            await cm.SetAsync(string.Format(Defaults.ShipModule.RedisPlayersPrefix, notification.Player.Email), notification.Player);
        }
        else
        {
            await cm.SetAsync(string.Format(Defaults.ShipModule.RedisPlayersPrefix, notification.Player.Email), notification.Player);
        }
    }
}
```

New Feature

Test Again

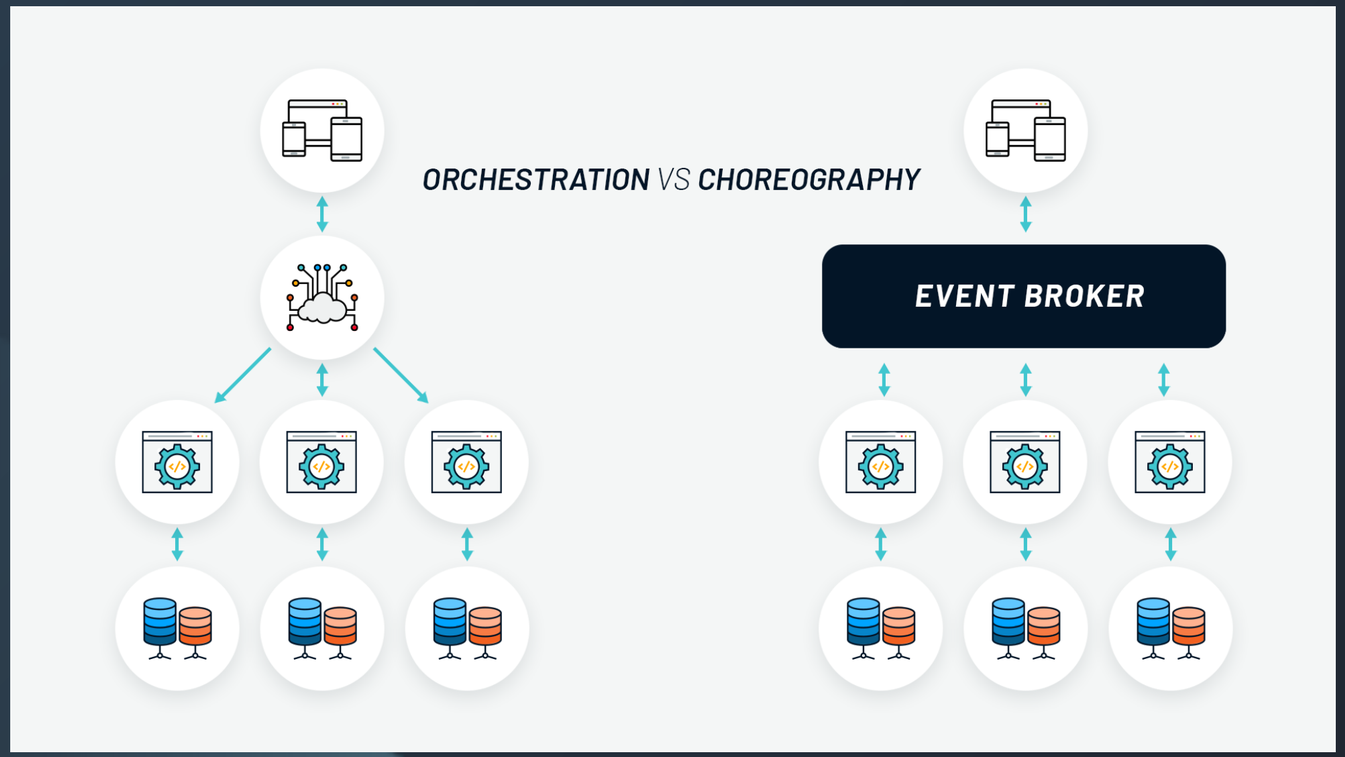
Saga Pattern
Distributed Data Transactions

Xelseor Consulting Group



Saga Pattern
Distributed Data Transactions

Xelseor
Consulting Group



Follow Me
& Download the Code!

Xelseor
Consulting Group

GitHub
Code

Linked In