1.Install MediatR nuget package.

builder.Services.AddMediatR(config => config.RegisterServicesFromAssemblies(typeof(Program).Assembly));

// Register MediatR services for the current assembly

2. Creating Mockdata MFakeDataStore.cs

builder.Services.AddSingleton<MFakeDataStore>();

// Register the MFakeDataStore as a singleton service

3.Create 3 folders Queries, Commands & Handlers

In Queries folder create a GetMProductQuery class

public record GetMProductQuery:IRequest<IEnumerable<MProduct>>;

In Handlers folder

public class GetMProductsHandler:

IRequestHandler<GetMProductQuery, IEnumerable<MProduct>>

{

private readonly MFakeDataStore \_dataStore;

public GetMProductsHandler(MFakeDataStore dataStore)

{

\_dataStore = dataStore;

}

public async Task<IEnumerable<MProduct>>

Handle(GetMProductQuery request, CancellationToken cancellationToken)

=> await \_dataStore.GetAllMProducts();

}

In Controller class

private readonly ISender \_sender;

public MProductController(ISender sender)

{

\_sender = sender;

}

[HttpGet]

public async Task<IActionResult> GetMProducts()

{

var products = await \_sender.Send(new GetMProductQuery());

return Ok(products);

}

4. We can create as many as classes in the Queries folder for example getproducts by ID

In Queries:

public record GetMProductbyIDQuery(int ID):IRequest<MProduct>;

In Handlers:

public class GetMProductByHandler:

IRequestHandler<GetMProductbyIDQuery, MProduct>

{

private readonly MFakeDataStore \_dataStore;

public GetMProductByHandler(MFakeDataStore dataStore)

{

\_dataStore = dataStore;

}

public async Task<MProduct> Handle(GetMProductbyIDQuery request, CancellationToken cancellationToken)

=> await \_dataStore.GetMProductbyID(request.ID);

}

In Controller:

[HttpGet("{ID}",Name ="GetMProductByID")]

public async Task<IActionResult> GetMProductByID(int ID)

{

var products = await \_sender.Send(new GetMProductbyIDQuery(ID));

return Ok(products);

}

5.For Commands

In Commands folder

public record AddMProductCommand(MProduct mProduct) : IRequest<MProduct>;

In Handlers folder

public class AddMProductHandler : IRequestHandler<AddMProductCommand,MProduct>

{

private readonly MFakeDataStore \_dataStore;

public AddMProductHandler(MFakeDataStore dataStore)

{

\_dataStore = dataStore;

}

public async Task<MProduct> Handle(AddMProductCommand request,

CancellationToken cancellationToken)

{

await \_dataStore.AddMProduct(request.mProduct);

return request.mProduct;

}

}

In Controller:

[HttpPost]

public async Task<IActionResult> AddMProduct([FromBody] MProduct product)

{

await \_sender.Send(new AddMProductCommand(product));

return StatusCode(201);

return CreatedAtRoute("GetMProductByID", new { ID = product.Id }, productToReturn);

}

\*\* If we want to return the created Item to the client then follow the highlighted yellow color

SINGLE REQUEST WILL BE HANDLED BY SINGLE HANDLER, IF WE WAN TO HANDLE A SINGLE REQUEST BY MULTIPLE HANDLERS THAT’S WHERE ***NOTIFICATION*** COMES INTO THE PICTURE

Same like Query & Command, we need to create Notification [MProductAddedNotification], instead IRequest it inherits INotification, we can create a multiple handlers [MEmailHandler, MCacheInvalidationHandler] now single notification/request will be handled by two handlers