Hexagonal Architecture

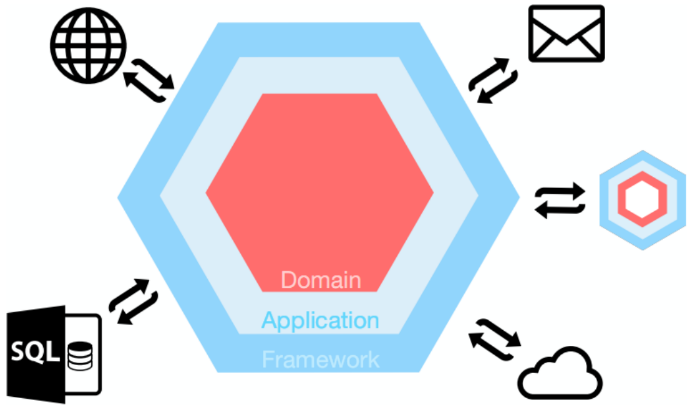
Other shapes

Ports and Adapters (different name)

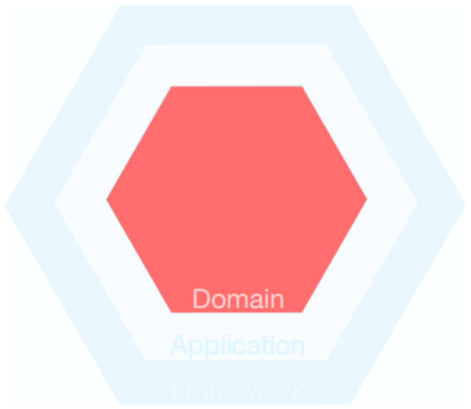
Onion Architecture

Clean Archtecture

Hexagonal Architecture



Domain Layer



Business rules

```
- Behaviors
- Constraints
- Invariants
```

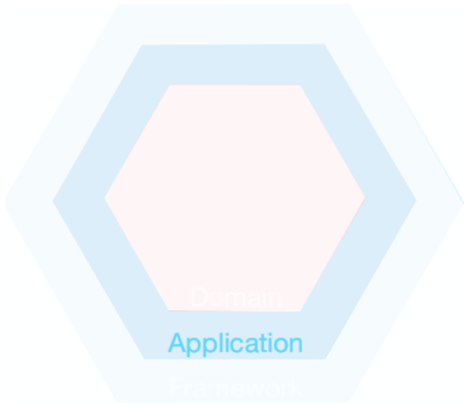
Domain Layer

```
public class Order
{
    private OrderId _id;
    private IList<OrderItem> _items = new List<OrderItem>();
    private Money sum = new Money(0, PLN);
    //.... status, createDate, rebatePolicy,

    public void Add(Product product, int quantity)
    {
         OrderItem oi = _orderItemFactory.Build(product, quantity, rebatePolicy);
         Items.add(oi);
         sum = sum.Add(oi.Cost);
     }

     public void Submit()
     {
         if (status != Status.NEW)
              throw new InvalidStateException();
          status = Status.SUBMITTED;
          _createDate = DateTime.Now ;
          eventsManager.Fire(new OrderSubmittedEvent(snapshot()));
     }

     public IList<OrderedProduct> OrderedItems()
     {
          return new List<OrderedProduct>(_items);
     }
}
```

Business rules

- Behaviors
- Constraints
- Invariants

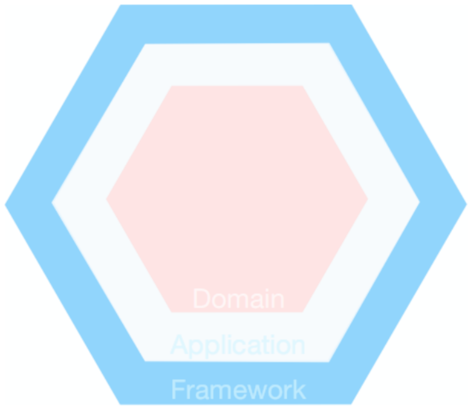
Application Layer



Application Layer

```
public class PurchaseService : IPurchaseService
{
  public string CreateOrder()
  {
    Order order = _orderFactory.Create(_clientRepository.Get(_systemUser.ClientNumber))
    return _orderRepository.Save(order).Number;
  }

  public void AddProduct(string orderNo, string productNo, int quantity)
  {
    Product product = _productsRepository.load(productNo);
    Order order = _ordersRepository.Load(orderNo);
    order.AddProduct(product, quantity);
    _ordersRepository.Save(order);
  }

  public void Submit(string orderNo, Payment payment)
  {
    Order order = _ordersRepository.Load(orderNo);
    order.Submit(payment);
    Invoice invoice = _bookKeeper.Issue(order);
    _ordersRepository.Save(order);
    _invoicesRepository.Save(invoice);
  }
}
```

Infrastructure Layer



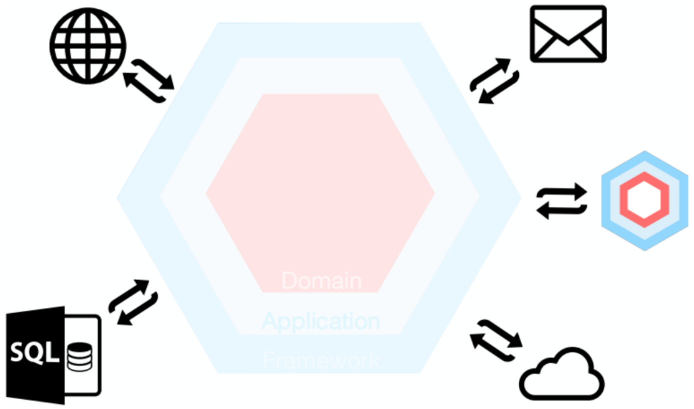
Infrastructure examples

```
- Repository implementation
- Queueing system access
- Email sending implementation
- ASP Net Core Controllers / WCF Sercices
```

Infrastructure Layer

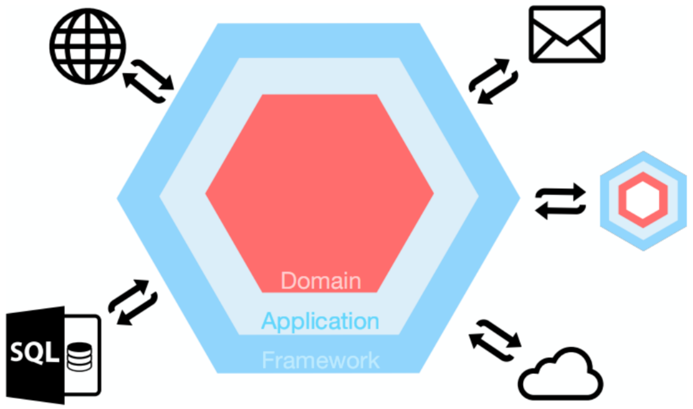
```
[Route("api/documents")]
[ApiController]
public class CreateDocumentController : Controller
{
  private readonly IDocumentService _service;

  public CreateDocumentController(IDocumentService documentService)
  {
    _service = documentService;
  }

  [HttpPost]
  public IActionResult PostCreateDocument([FromBody] PostCreateDocumentRequest request)
  {
    var id = Guid.NewGuid().ToString();

    _service.CreateDocument(new CreateDocumentCommand(
      id,
      request.CreatorId,
      Enum.Parse<DocumentType>(request.DocumentType),
      request.Title));

    return Created("api/documents", id);
  }
}
```

Externals



- databases
- emails
- other services
- other hexagons
- file system

Ports & Adapters



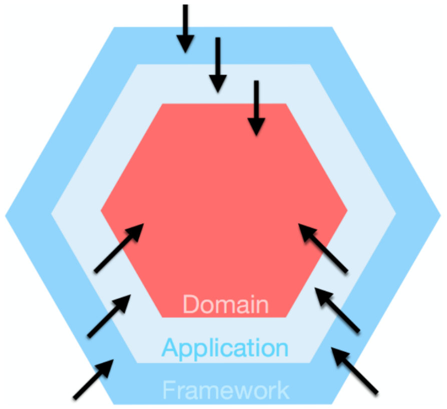
Why "Ports & Adapters"

- Adapter
- Port

Communication

- Outside in (**command**)
- Inside out (**event**)

Dependecies



Communication

- Outside in (**command**)
- Inside out (**event**) - IoC

Live demo

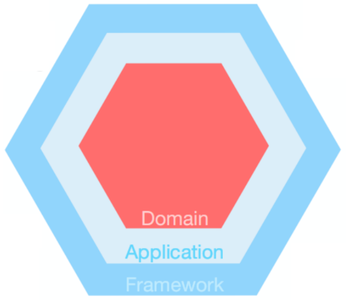
Testing Hexagon

How to test ?

- Domain layer (no dependencies)
- Application layer (limited ifs)
- Infrastructure / integration (limited ifs)

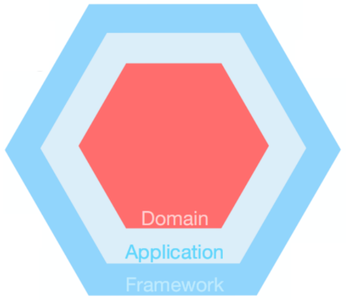
Layers vs Test Coverage

Checking questions



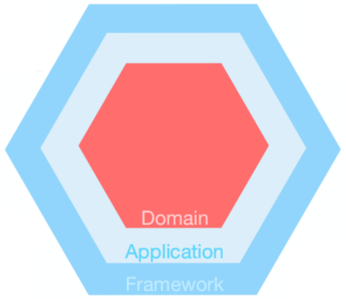
Checking questions

- Where should the validation be placed?



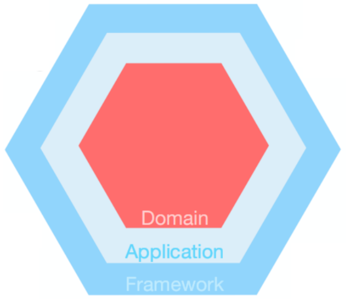
Checking questions

- Where should the validation be placed?
- How does the domain communicate with the outside world?



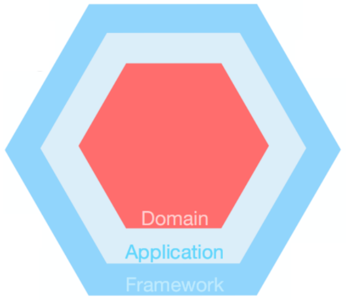
Checking questions

- Where should the validation be placed?
- How does the domain communicate with the outside world?
- Examples of stuff outside the domain



Checking questions

- Where should the validation be placed?
- How does the domain communicate with the outside world?
- Examples of stuff outside the domain
- How can I log something in domain layer? (NLog, log4net)



Exercise: Hexagon

Sort out files to correct projects in a hexagon solution.
Each file should be placed in correct layer of the hexagon

Hexagonal Architecture (wrap up)

- Domain Layer (Behaviors, invariants, rules, constraints)
- Application layer (orchestration)
- Infrastructure layer (ASP. Mvc)
- Dependency direction

Hexagonal architecture (sources)

Articles

- Alistair Cockburn - Hexagonal Architecture (http://alistair.cockburn.us/Hexagonal+architecture)

Wzorce Biznesowe

Repositories

Repositories

Abstraction of data source. It provides access to Entities and Aggregates

- data sources – can be many
- Interface belongs to domain
- Repository must inject dependencies to aggregates
- Provides optimistic locking of entire aggregate

It manages loading and storing of objects

- Can load Aggregate by Id or business criteria
- don't use them as a query mechanism

Repositories (Quiz)

Get	null
Load	IEnumerable
Find	Exception

Repositories

```
public interface DocumentRepository
{
  public Document Get(id);
  public void Save(Document);
  public void Delete(Document);

  public List<Document> Find(Criteria criteria);
}
```

Repositories - dependency injection #1

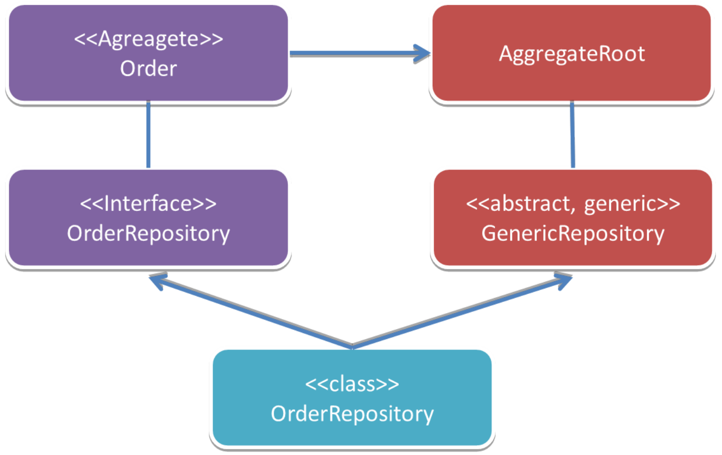
```
public class DocumentRepository : IDocumentRepository
{
  private IWindsorContainer _container;

  public DocumentRepository(IWindsorContainer container)
  {
    _container = container;
  }

  public Document Get(Guid id)
  {
    Document result = _session.Get<Document>(id);
    result.SetEventPublisher(_container.Resolve<IEventPublisher>());
    return result;
  }
}
```

Repositories - dependency injection #2

```
public class DocumentRepository : IDocumentRepository
{
  private IDependencyInjector _dependencyInjector;

  public DocumentRepository(
    IDependencyInjector dependencyInjector)
  {
    _dependencyInjector = dependencyInjector;
  }

  public Document2 Get(Guid id)
  {
    Document2 result = _session.Get<Document2>(id);
    _dependencyInjector.InjectDependencies(result);
    return result;
  }
}
```

```
public class DependencyInjector : IDependencyInjector
{
  private readonly IWindsorContainer _container;

  public DependencyInjector(IWindsorContainer container)
  {
    _container = container;
  }

  public void InjectDependencies(AggregateRoot aggregateRoot)
  {
    var fields = aggregateRoot.GetType().GetFields();
    foreach (FieldInfo fieldInfo in fields)
    {
      if (fieldInfo.FieldType.IsInterface)
      {
        object iface = _container.Resolve(fieldInfo.FieldType);
        fieldInfo.SetValue(aggregateRoot, iface);
      }
    }
  }
}
```

GenericRepository



<<Agreagete>> Order
AggregateRoot
<<Interface>> OrderRepository
<<abstract, generic>> GenericRepository
<<class>> OrderRepository

GenericRepository (NHibernate)

```
public class GenericRepository<TAggregateRoot> : IGenericRepository<TAggregateRoot>
        where TAggregateRoot : AggregateRoot
{
  private readonly IDependencyInjector _dependencyInjector;
  private readonly ISession _session;

  public GenericRepository(ISession session, IDependencyInjector dependencyInjector)
  {
    _session = session;
    _dependencyInjector = dependencyInjector;
  }

  public TAggregateRoot Load(AggregateId id)
  {
    var result = _session.Get<TAggregateRoot>(id);
    _dependencyInjector.InjectDependencies(result);
    return result;
  }

  public void Save(TAggregateRoot aggregateRoot)
  {
    _session.SaveOrUpdate(aggregateRoot);
  }

  public void Delete(AggregateId id)
  {
    _session.Delete(_session.Get<TAggregateRoot>(id));
  }

}
```

GenericRepository (EF Core)

```
public class GenericRepository<T> : IGenericRepository<T>
  where T : AggregateRoot
{
  protected readonly GenericContext<T> _context;
  protected readonly IEventBus _eventBus;

  public GenericRepository(
    GenericContext<T> context,
    IEventBus eventBus)
  {
    _context = context;
    _eventBus = eventBus;
  }

  public T Get(string id)
  {
    var item = _context.Items.First(f => f.Id == id);
    (item as IDependencySetter).SetEventPublisher(_eventBus);
    return item;
  }

  public virtual void Save(T obj)
  {
    if (_context.Entry(obj).State == EntityState.Detached)
      _context.Items.Add(obj);

    _context.SaveChanges();
  }
}
```

```
public class GenericContext<T> : DbContext where T : class
{
  private readonly string _connectionString;
  public DbSet<T> Items { get; set; }


  public GenericContext(string connectionString)
  {
    _connectionString = connectionString;
  }

  protected override void OnConfiguring(
    DbContextOptionsBuilder optionsBuilder)
  {
    optionsBuilder.UseSqlServer(_connectionString);
  }
}
```

Mapping (NHibernate)

```
public class OrderMap : ClassMapping<Order>
{
  public OrderMap()
  {
    Id(x => x.Id, m =>
    {
      m.Column("Id");
      m.Generator(Generators.Identity);
    });
    Property(x => x.Status);
    Property(x => x.ClientId);
    Property(x => x.SubmintDate);
    //...
  }
}
```

Mapping (EF Core)

```
public class OrderContext : GenericContext<Order>
{
  public OrderContext(string connectionString) : base(connectionString)
  {
  }

  protected override void OnModelCreating(ModelBuilder modelBuilder)
  {
    modelBuilder.Entity<Order>(c =>
    {
      c.ToTable("Order");
      c.HasKey("_id");
      c.Property("_id").HasColumnName("id");
      c.Property("_number").HasColumnName("number");
      c.HasMany<OrderItem>("_products");
    });
    modelBuilder.Entity<OrderItem>(c =>
    {
      c.ToTable("OrderItem");
      c.HasOne(x => x.Order).WithMany("_products").HasForeignKey("OrderId");
      c.Property<string>(f => f.ProductId);
    });
  }
}
```

EF /EF.Core

https://stackoverflow.com/questions/7619955/mapping-private-property-entity-framework-code-first
https://csharp.christiannagel.com/2016/11/07/efcorefields/

Persistency VO (NHibernate)

```
public class ClientMap : ClassMapping<Client>
{
  public ClientMap()
  {
    //...

    Property("_taxNumber", m =>
    {
      m.Type(new ValueTypeAsStringType<TaxNumber>());
    });

    Component<Address>("_address",
      m =>
      {
        m.Property(x => x.AddressLine1);
        m.Property(x => x.AddressLine2);
      });
  }
  ...
}
```

Persistency VO (EF Core)

```
public class OrderContext : DbContext
{
  private string _connectionString;

  public OrderContext(string connectionString)
  {
    _connectionString = connectionString;
  }

  protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
  {
    optionsBuilder.UseSqlServer(_connectionString);
  }

  protected override void OnModelCreating(ModelBuilder modelBuilder)
  {
    modelBuilder.Entity<Order>(c =>
    {
      c.ToTable("Order");
      c.HasKey("_id");
      c.OwnsOne<Money>("_price", f =>
        {
          f.Property("_amount").HasColumnName("net_amount");
          f.Property("_currency").HasColumnName("net_currency");
        });

    });
  }
}
```

Memento

```
public class Document :
  AggregateRoot, IStateAccesor<DocumentState>
{
  private DocumentState _state;

  public Document(DocumentNumber documentNumber)
  {
    _state = new DocumentState();
    Number = documentNumber;
  }

  public Document(DocumentState state)
  {
    _state = state;
  }

  private DocumentNumber Number
  {
    get { return _state.Number; }
    private set { _state.Number = value; }
  }


  DocumentState IStateAccesor<DocumentState>.GetState()
  {
    return _state;
  }
}
```

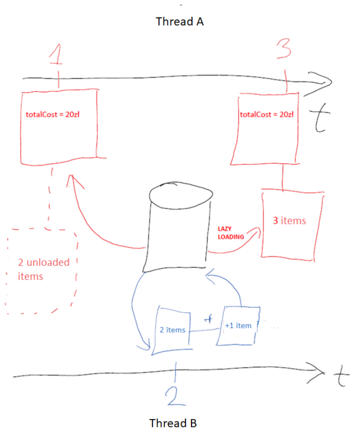
```
public class DocumentStateMap : ClassMapping<DocumentState>
{
  public DocumentStateMap()
  {
    Lazy(false);

    Id(x => x.Id);
    Property(x => x.Number);
  }
}
```

```
public class Repository
{
  public void Save(Document document)
  {
    _session.SaveOrUpdate(
      (document as IStateAccesor).GetState())
  }

  public Document Load(Guid id)
  {
    Document result
      = new Document(_session.Get<DocumentState>(id);
    ...
  }
}
```

[illegible]

Lazy Loading or Eager Loading ?

Traps of Lazy Loading'u



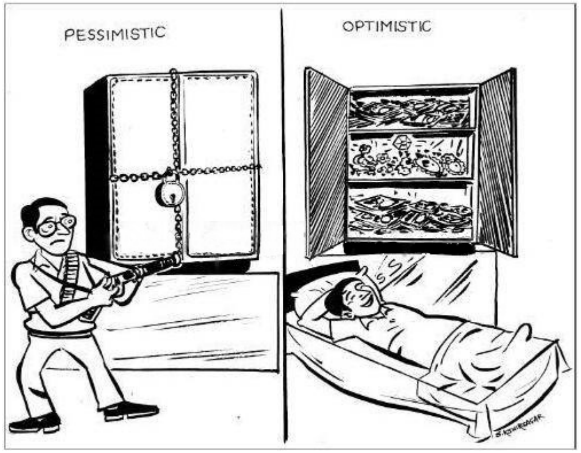
Problem

- Thread A loads data (order without items)
- Thread B adds new item to order
- Thread A loads through LAZY Loading rest of data
  .... BUM !!!
- TotalCost is different than sum of items

Solution:

- EAGER Loading
- Calculate total cost when used (what if it's costly ?)

Locking



Kinds of locking

Optimistic

- Concurrency is an exception
    - We assume that loaded aggregate will not change
    - .... if not EXCEPTION !!!
- Implementation can be based on Version field or Timestamp dependent on database. What if you have multiple servers ?
- Cons - exceptions

Pessimistic

- We assume that concurrency will appear
- So we create lock during reading
- Implemented through database
- Cons - performance

Optimistic locking (NHibarnate)

C#

```
public class VersionedEntityClassMapping<VersionedEntity>
{
  public VersionedEntityClassMapping()
  {
    Id(x => x.Id, x => x.Generator(Generators.Identity));

    Version(x => x.Version, x => x.Type(NHibernateUtil.Int32));

    OptimisticLock(OptimisticLockMode.Version);
    //NHibernate 4.1 only, no need to specify since this is the default
   }
}
```

SQL

```
Update [Order] Set Name = ’DW/123’, Version = 2
where Id = 1 and Version = 1
```

https://ayende.com/blog/3946/nhibernate-mapping-concurrency

Optimistic locking (Entity Framework)

```
public class Department
{
    ...

    [Timestamp]
    public byte[] RowVersion { get; set; }

    ...
}
```

Fluent Api

```
modelBuilder.Entity<Department>()
    .Property(p => p.RowVersion).IsConcurrencyToken();

Property(p => p.RowVersion).IsRowVersion();
```

Pessimistic locking

C#

```
using (var session = sessionFactory.OpenSession())
using (var tx = session.BeginTransaction())
{
  var person = session.Get<Person>(1,LockMode.Upgrade);
  person.Name = "other";
  tx.Commit();
}
```

SQL

```

select Id, Name
from [Order] with(rowlock, updlock)
where Id = 1
```

What if we add items to order, what will change ?

What if we add items to order, what will change ?

```
class Order
{
  int _version;
  List<OrderItem> _reservationItems;

  public void Add(Product product)
  {
    _reservationItems.Add(new OrderItem(product.Id))
    _version++;
  }
}
```

CommandHandler

Example service implementation WebApi

```
[Route("api/[controller]")]
[ApiController]
public class DocumentsController : ControllerBase
{
  IDocumentService _documentService

  public ApiController(IDocumentService documentService)
  {
     _documentService = documentService;
  }

  [HttpPost()]
  public IActionResult Add(...)
  {
    _documentService.CreateDocument(...)
  }

  [HttpPost("{id}/confirmations")]
  public IActionResult Confirm(...)
  {
    _documentService.ConfirmDocument(...)
  }
}
```

```
public class DocumentService : IDocumentService
{
  IDocumentRepository _repository;

  public DocumentService(IDocumentRepository repository)
  {
    _repository = repository
  }

  public int CreateDocument(...)
  {
    ...
  }

  public void ConfirmDocument(...)
  {
    ...
  }
}
```

CommandHandler implementation

```
[Route("api/[controller]")]
[ApiController]
public class DocumentsController : ControllerBase
{
  ICommandHandler<CreateDocumentCommand> _createDocumentHandler

  public ApiController(
      ICommandHandler<CreateDocumentCommand> createDocumentHandler)
  {
     _createDocumentHandler = createDocumentHandler;
  }

  [HttpPost()]
  public IActionResult Add(...)
  {
    _createDocumentHandler.Handle(...)
  }
}
```

```
public interface ICommandHandler<TCommand>
{
  void Handle(TCommand command);
}

public class CreateDocumentHandler :
    ICommandHandler<CreateDocumentCommand>
{
  public void Handle(...)
  {
    ...
  }
}
```

CommandHandler implementation

```
[Route("api/[controller]")]
[ApiController]
public class DocumentsController : ControllerBase
{
  ICommandHandler<CreateDocumentCommand> _createDocumentHandler

  public ApiController(
      ICommandHandler<CreateDocumentCommand> createDocumentHandler)
  {
     _createDocumentHandler = createDocumentHandler;
  }

  [HttpPost()]
  public IActionResult Add(...)
  {
    _createDocumentHandler.Handle(...)
  }
}
```

```
public interface ICommandHandler<TCommand>
{
  void Handle(TCommand command);
}

public class CreateDocumentHandler :
    ICommandHandler<CreateDocumentCommand>
{
  public void Handle(...)
  {
    ...
  }
}
```

Cohesion ??

What aspect should be covered in CommandHandlers

- Common interface
  - Security
  - Logging
- Mapping outside messages to internal operations and processes
- Communication with services in the Domain and Infrastructure layers to provide cohesive operations for outside clients
- Business rules are not allowed !!!

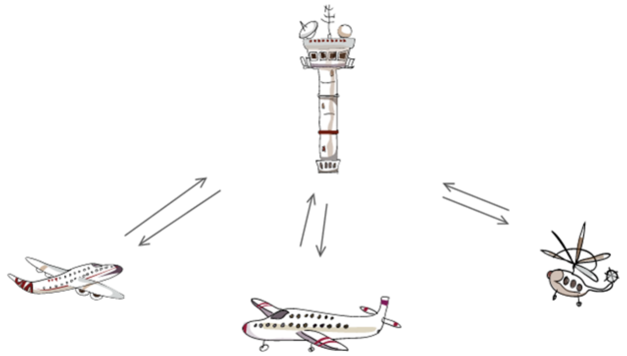
```
public interface ICommandHandler<TCommand>
{
  void Handle<TCommand>(TCommand command);
}
```

```
public class CalculateFraudProbabilityCommandHandler
  : ICommandHandler<CalculateFraudProbabilityCommand>
{
  //Execute Command
  public void Handle(CalculateFraudProbabilityCommand command)
  {
    Customer customer = _customerRepository.GetById(command.CustomerId);
    FraudHistory fraudHistory = _fraudService.RetrieveFraudHistory(customer);

    //any fraud recently? if so, let someone know!
    if(fraudHistory.FraudSince(DateTime.Now.AddYear(-1))
    {
      _notifier.SendEmail(_fraudService.BuildFraudWarningEmail(
        customer,
        fraudHistory));
    }
  }
}
```

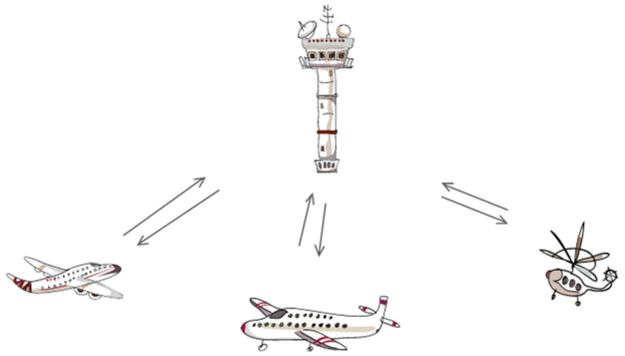
Live coding: Command Handler

Mediator

Mediator

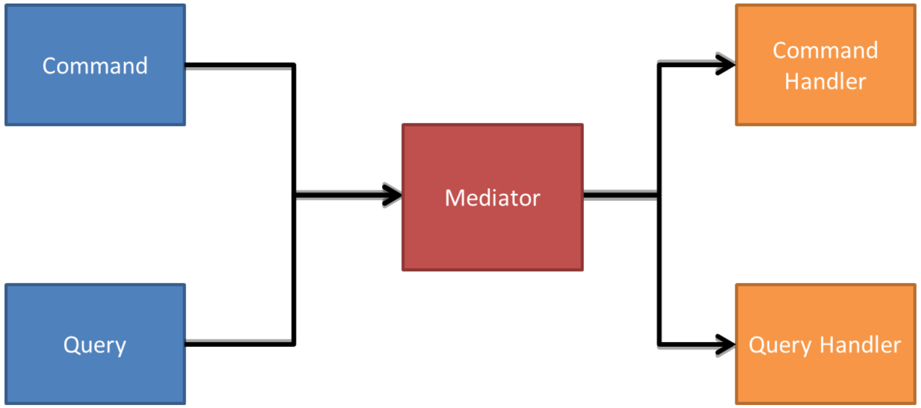


Mediator



What would happen if there was no control tower?

Mediator



Mediator - typical approach 1

```
public class ClientController : Controller
{
  IClientService _clientService;

  public ClientController(IClientService clientService)
  {
    clientService = _clientService;
  }

  public ActionResult Add(AddClientModel client)
  {
    _clientService.Add(new AddClientCommand(client))
  }

  public ActionResult Update(UpdateClientModel client)
  {
    _clientService.Update(new UpdateClientCommand(client))
  }
}
```

Mediator - typical approach 2

```
public class ClientController : Controller
{
  IAddClientHandler _addClientHandler;
  IUpdateClientHandler _updateClientHandler;

  public ClientController(
    IAddClientHandler addClientHandler,
    IUpdateClientHandler updateClientHandler)
  {
    _addClientHandler = addClientHandler;
    _updateClientHandler = updateClientHandler;
  }

  public ActionResult Add(AddClientModel client)
  {
    _addClientHandler.Handle(new AddClientCommand(client))
  }

  public ActionResult Update(UpdateClientModel client)
  {
    _updateClientHandler.Handle(new UpdateClientCommand(client))
  }
}
```

Mediator - sender

```
public class ClientController : Controller
{
  IMediator _mediator;

  public ClientController(IMediator mediator)
  {
    _mediator = mediator
  }

  public ActionResult Add(AddClientModel client)
  {
    _mediator.Send(new AddClientCommand(client))
  }

  public ActionResult Update(UpdateClientModel client)
  {
    _mediator.Send(new UpdateClientCommand(client))
  }
}
```

Mediator - implementation

```
public class Mediator : IMediator
{
  private Dictionary<Type, IHandler> _handlers = new Dictionary<Type, IHandler>();

  void IMediator.Register<T>(IHandler handler)
  {
    _handlers.Add(typeof(T), handler);
  }

  void IMediator.Send<T>(T message)
  {
    if(_handlers.ContainsKey(typeof(T)))
    {
      _handlers[typeof(T)].Handle(message)
    }
    else
    {
      throw new InvalidOperation("unknown message type");
    }
  }
}
```

Mediator - implementation autofac

```
public class Bus : ICommandBus, IEventBus
{
  private readonly IComponentContext _container;

  public Bus(IComponentContext container)
  {
    _container = container;
  }

  // ICommandBus.Send
  public void Send<T>(T command)
  {
    ICommandHandler<T> commandHandler = (ICommandHandler<T>)_container.Resolve(typeof(ICommandHandler<T>));
    commandHandler.Handle(command);
  }

  // IEventBus.Publish
  public void Publish<T>(T @event)
  {
    IEnumerable<IEventHandler<T>> eventHandlers =
      (IEnumerable<IEventHandler<T>>) _container.Resolve(typeof(IEnumerable<IEventHandler<T>>));

    foreach (var eventHandler in eventHandlers)
    {
      eventHandler.Handle(@event);
    }
  }
}
```

Mediator - receiver

```
public class AddClientMessageHandler : ICommandHandler<AddClientCommand>
{
  public void Handle(AddClientCommand command)
  {
    ...
  }
}

public class ClientAddressUpdatedHandler : IEventHandler<ClientAddressUpdatedEvent>
{
  public void Handle(ClientAddressUpdatedEvent command)
  {
    ...
  }
}
```

Mediator - receiver

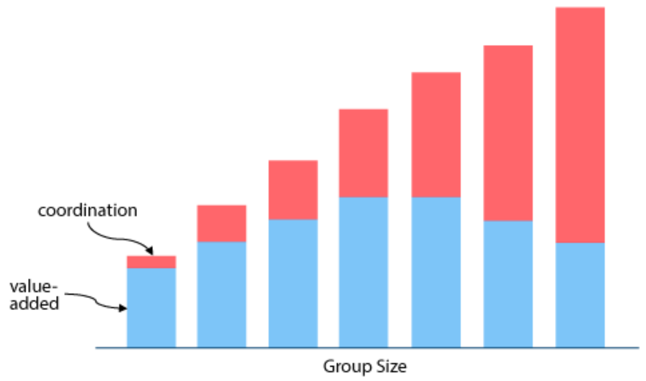
```
public class AddClientMessageHandler : ICommandHandler<AddClientCommand>
{
  public void Handle(AddClientCommand command)
  {
    ...
  }
}

public class ClientAddressUpdatedHandler : IEventHandler<ClientAddressUpdatedEvent>
{
  public void Handle(ClientAddressUpdatedEvent command)
  {
    ...
  }
}
```

MediatR – Jimmie Bogard - https://github.com/jbogard/MediatR

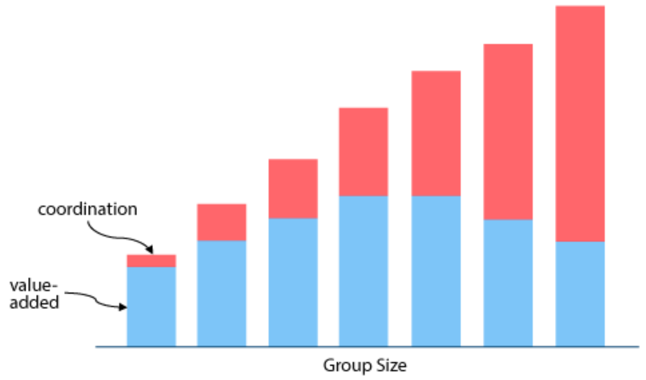
Example: Hexagon + CQRS + Mediator + CommandHandler

Vertical Slices

Have you ever worked in 100+ people on one product?

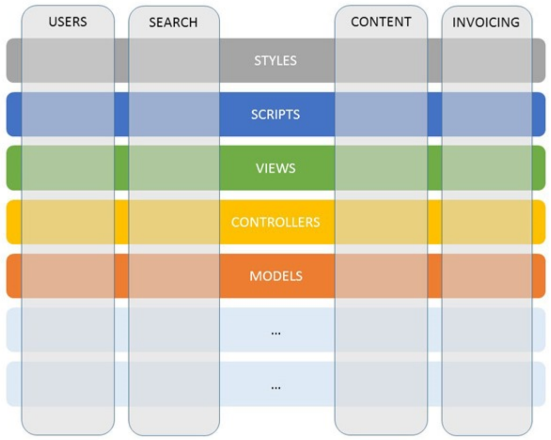


Have you ever worked in 100+ people on one product?



Delayed project - let's add 2 more teams !!!

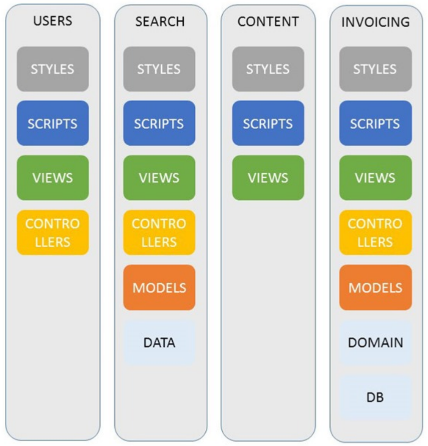
Horizontal Slices



Design functionalities so that the cost of removal is minimal

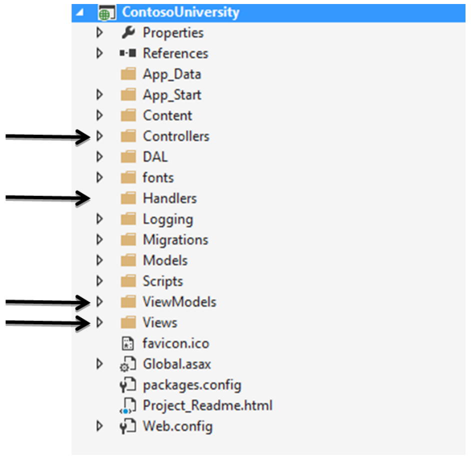
- Independent functionalities on each other
- All (technical) elements are grouped together

Vertical Slices



Vertical Slices (Asp .Net MVC)

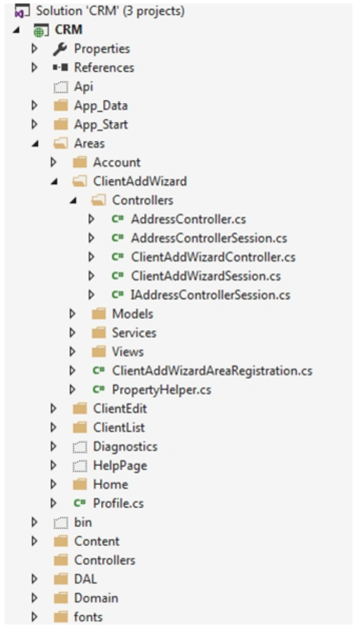
What is the structure of your project?



```
ContosoUniversity
  Properties
  References
  App_Data
  App_Start
  Content
  Controllers
  DAL
  fonts
  Handlers
  Logging
  Migrations
  Models
  Scripts
  ViewModels
  Views
  favicon.ico
  Global.asax
  packages.config
  Project_Readme.html
  Web.config
```

Area

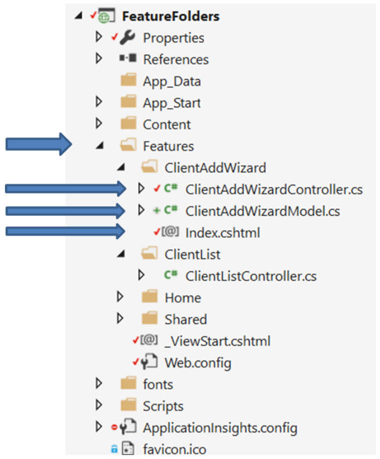
- What does it do
- Registration
- [ActionLinkArea]



```
Solution 'CRM' (3 projects)
  CRM
    Properties
    References
    Api
    App_Data
    App_Start
    Areas
      Account
      ClientAddWizard
        Controllers
          AddressController.cs
          AddressControllerSession.cs
          ClientAddWizardController.cs
          ClientAddWizardSession.cs
          IAddressControllerSession.cs
        Models
        Services
        Views
        ClientAddWizardAreaRegistration.cs
        PropertyHelper.cs
      ClientEdit
      ClientList
      Diagnostics
      HelpPage
      Home
      Profile.cs
    bin
    Content
    Controllers
    DAL
    Domain
    fonts
```

Feature Folders

- Where are the controllers?
- Where are the views?
- Where are the models?
- Cons?

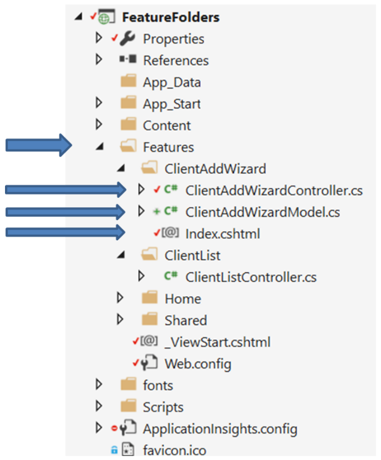


```
FeatureFolders
  Properties
  References
  App_Data
  App_Start
  Content
  Features
    ClientAddWizard
      ClientAddWizardController.cs
      ClientAddWizardModel.cs
      Index.cshtml
    ClientList
      ClientListController.cs
    Home
    Shared
    _ViewStart.cshtml
    Web.config
  fonts
  Scripts
  ApplicationInsights.config
  favicon.ico
```

Feature Folders

- Where are the controllers?
- Where are the views?
- Where are the models?
- Cons?

Demo !!!



```
FeatureFolders
  Properties
  References
  App_Data
  App_Start
  Content
  Features
    ClientAddWizard
      ClientAddWizardController.cs
      ClientAddWizardModel.cs
      Index.cshtml
    ClientList
      ClientListController.cs
    Home
    Shared
    _ViewStart.cshtml
    Web.config
  fonts
  Scripts
  ApplicationInsights.config
  favicon.ico
```

Resources (Vertical Slices)

Presentations

- SOLID Architecture in Slices not Layers - Jimmy Bogard (https://vimeo.com/131633177)
- Feature Folder Structure in ASP.NET Core - https://scottsauber.com/2016/04/25/feature-folder-structure-in-asp-net-core/