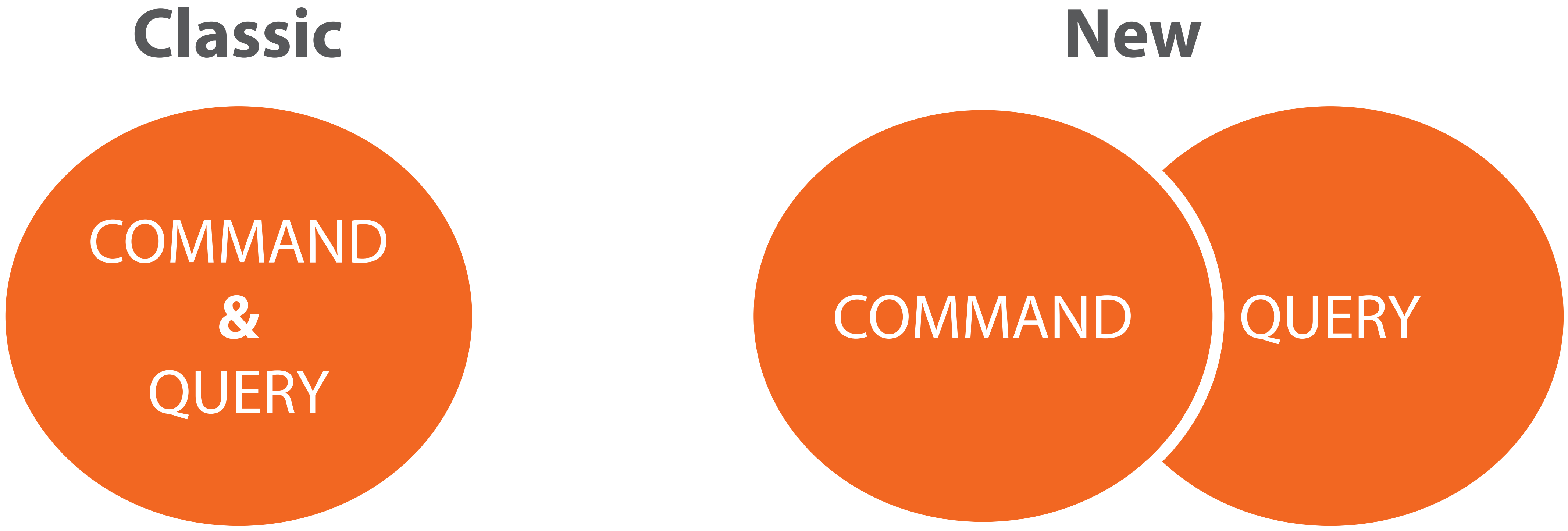
The CQRS Supporting Architecture

Dino Esposito

@despos | http://software2cents.wordpress.com

All That Everybody Wanted Was **Software Model for the Business Domain**



Key Points

CQRS	CQRS	CQRS
Regular	**Premium**	**Deluxe**

One

Comprehensive

Software Model

for the Business

Domain

...Optical Illusion...

Great for **commands**

```
public class Match
{
   public Match( ... ) { ... }
```

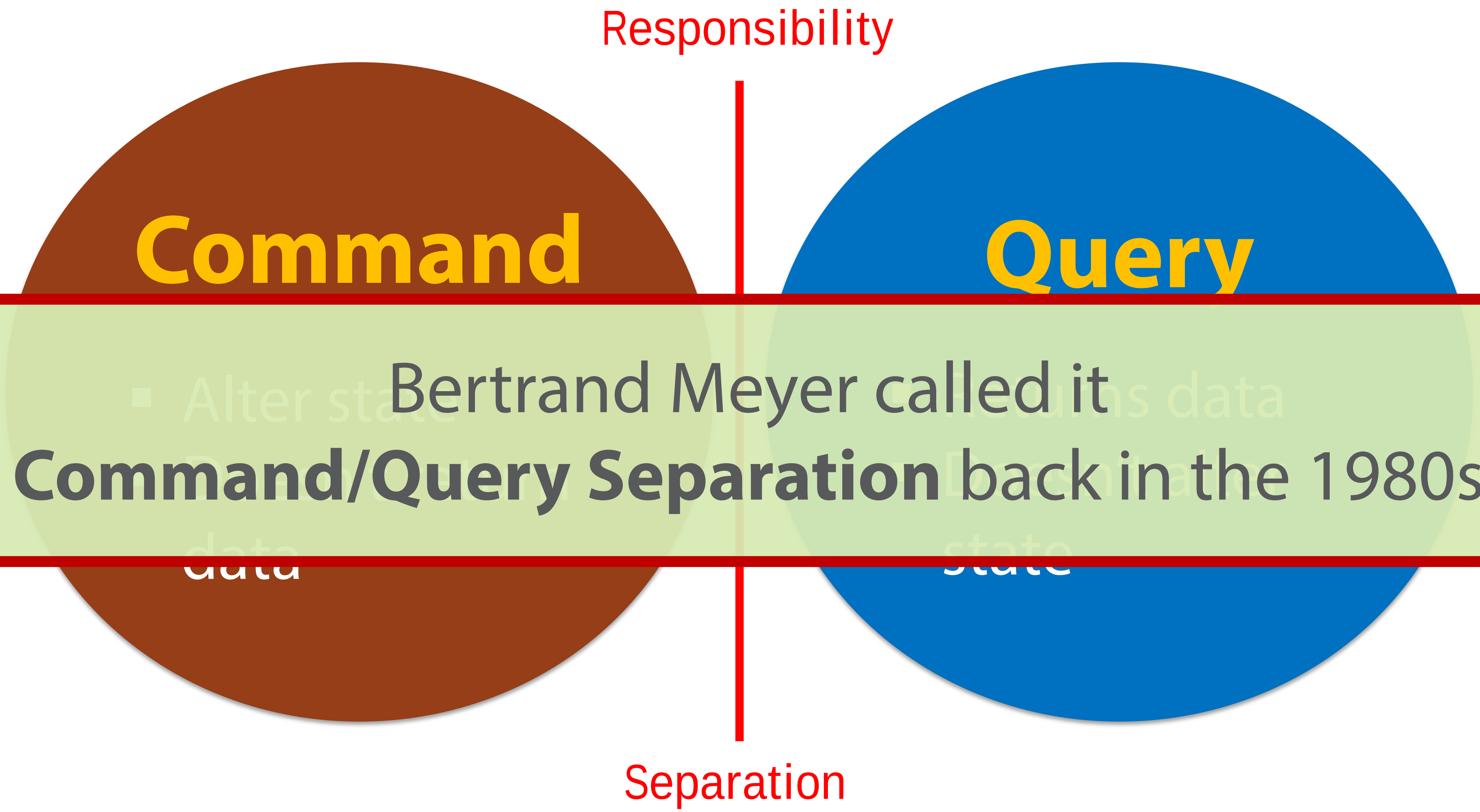
- Requires fixes for persistence
- Exposes behavior to presentation

Great for **queries**

```
public class Match
{
   public Score Score { get; set; }
   public int Period { get; set; }
```

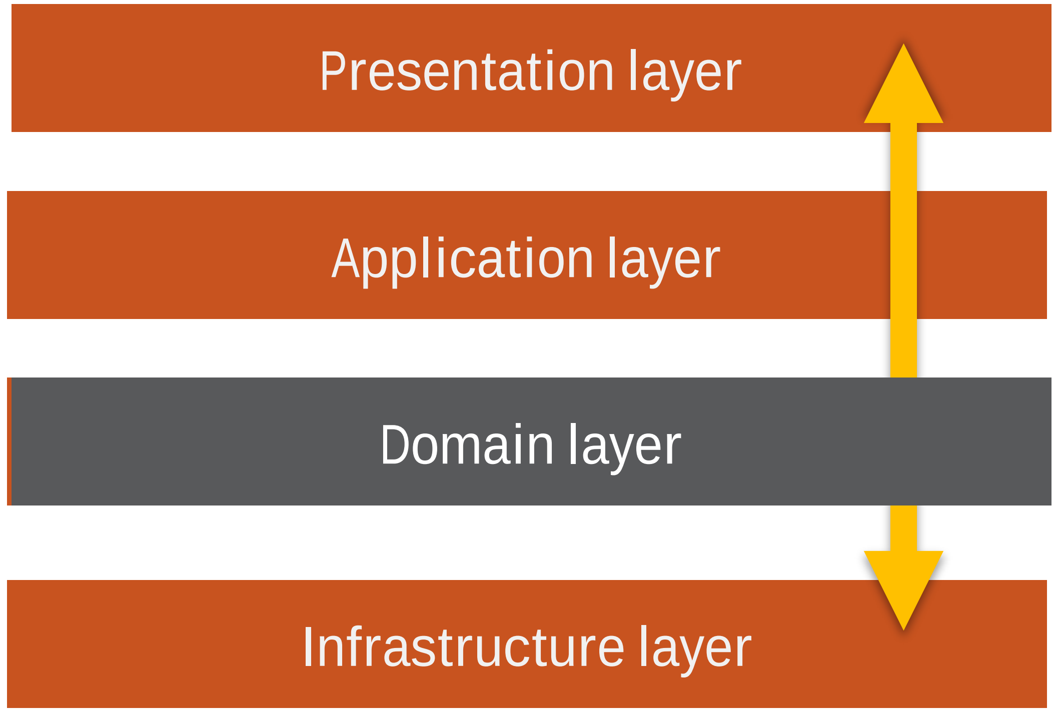
- No business rules in the class
- Risk of getting into incongruent state

CQRS

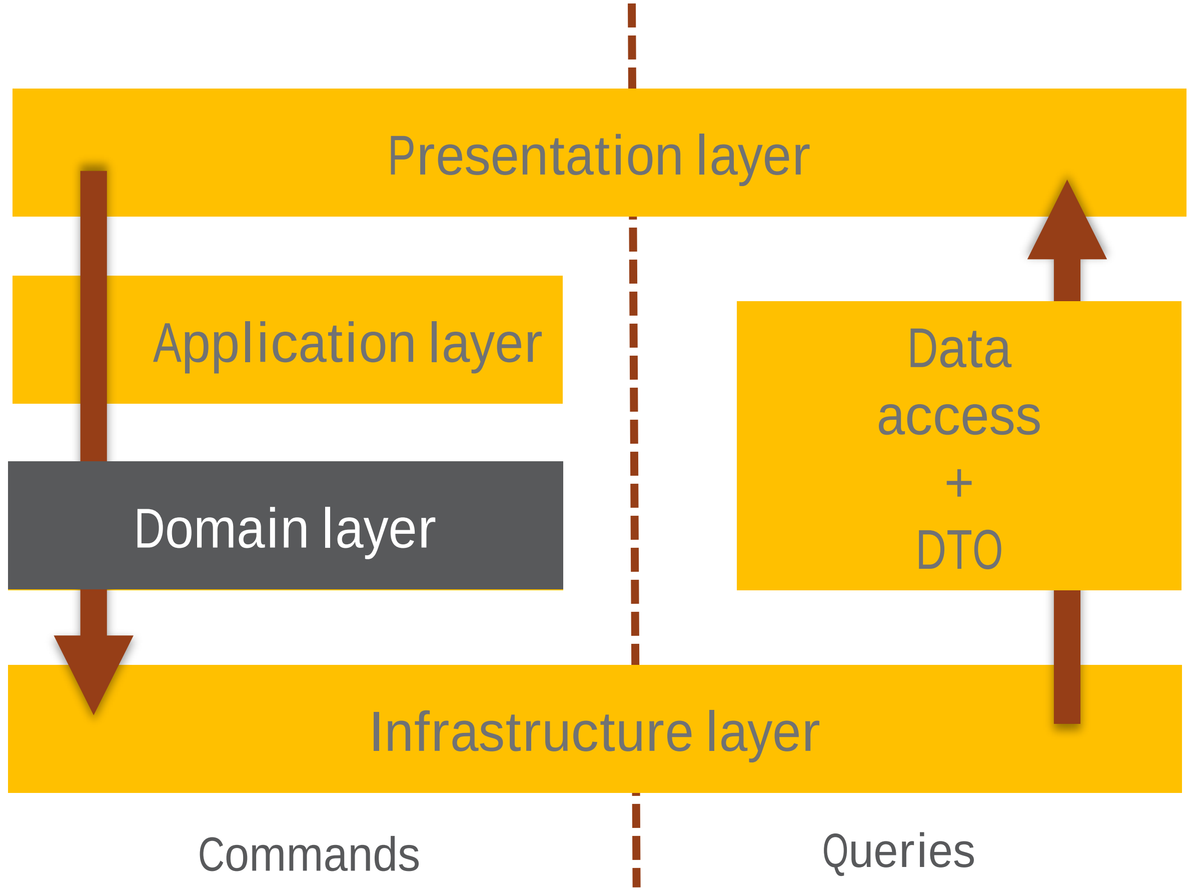


Bertrand Meyer called it
Command/Query Separation back in the 1980s

Canonical **Layered Architecture**



CQRS



Aspects of **CQRS**

Benefits

- Distinct **optimization**
- **Scalability** potential

Side effects

- **Simplified** design
- Hassle-free stacks **enhancement**

Flavors of CQRS

Regular
Premium
Deluxe

One

Comprehensive

Software Model

for the Business

Domain

...Optical Illusion...

Great for **commands**

```
public class Match
{
   public Match( ... ) { ... }
```

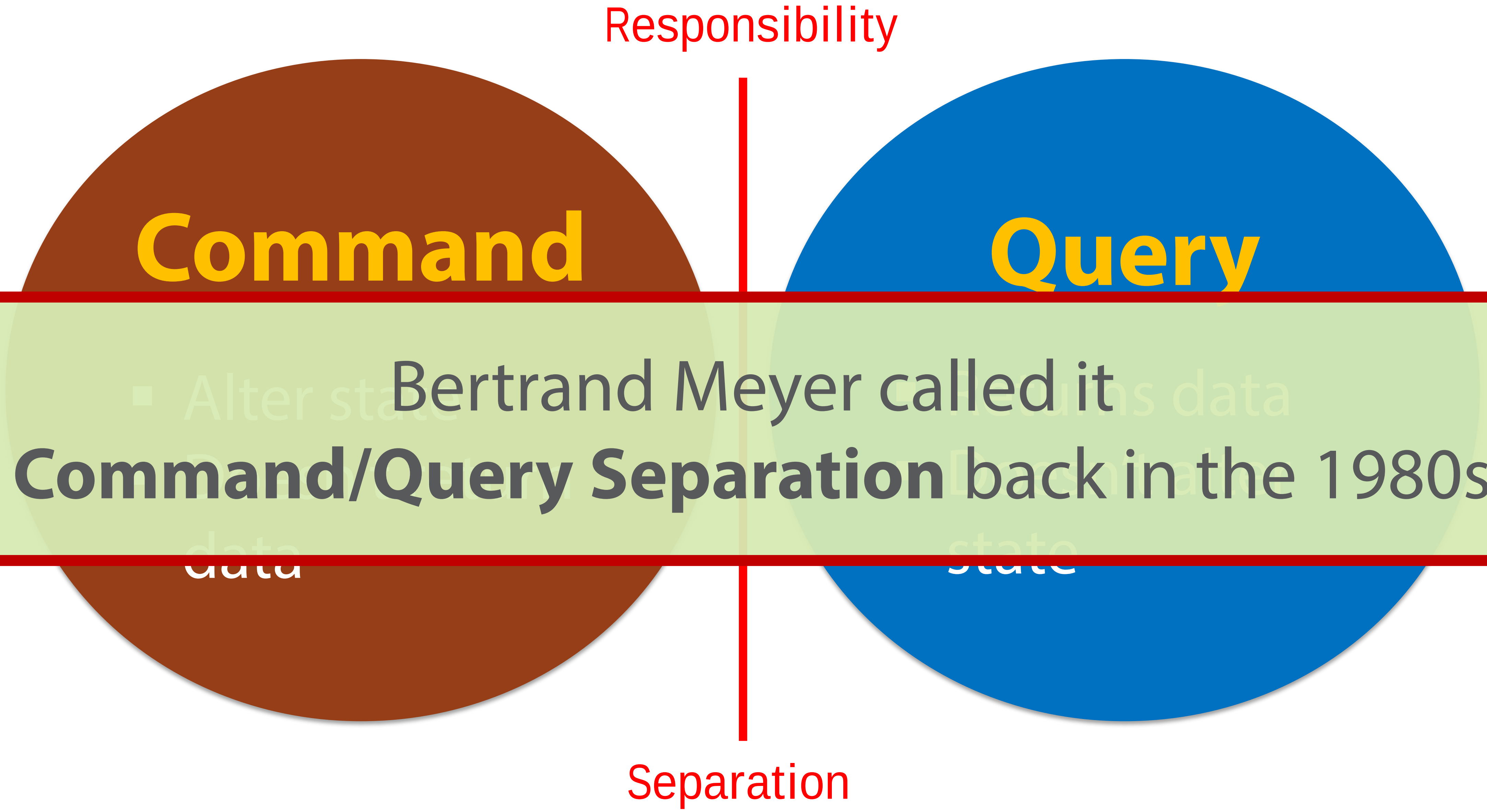
- Requires fixes for persistence
- Exposes behavior to presentation

Great for **queries**

```
public class Match
{
   public Score Score { get; set; }
   public int Period { get; set; }
```

- No business rules in the class
- Risk of getting into incongruent state

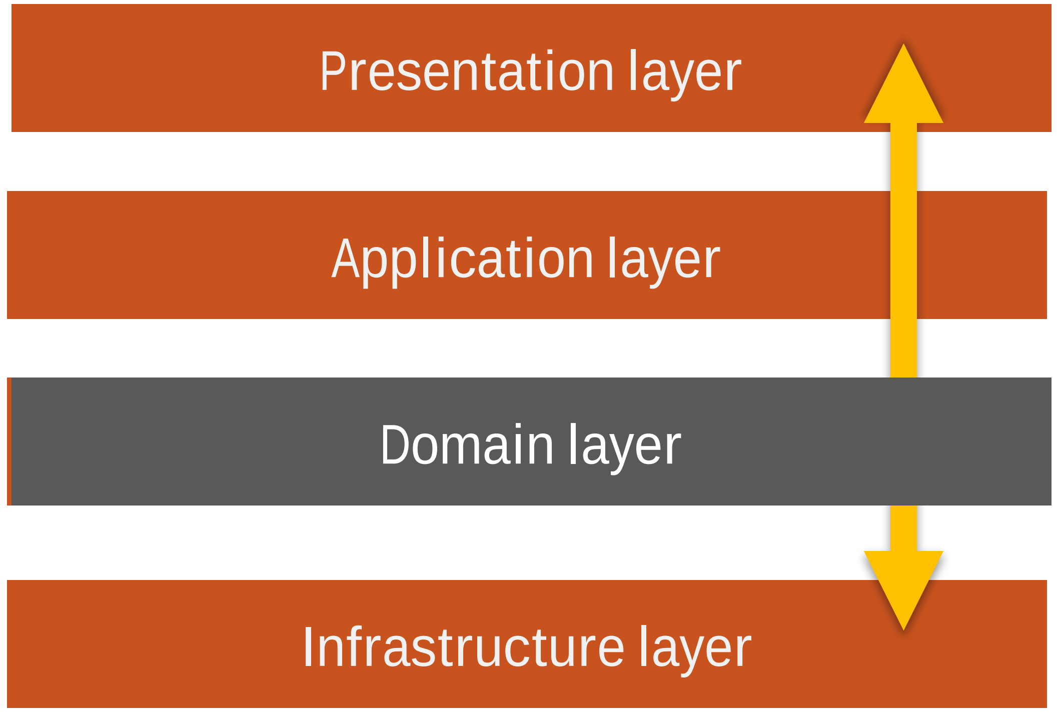
CQRS

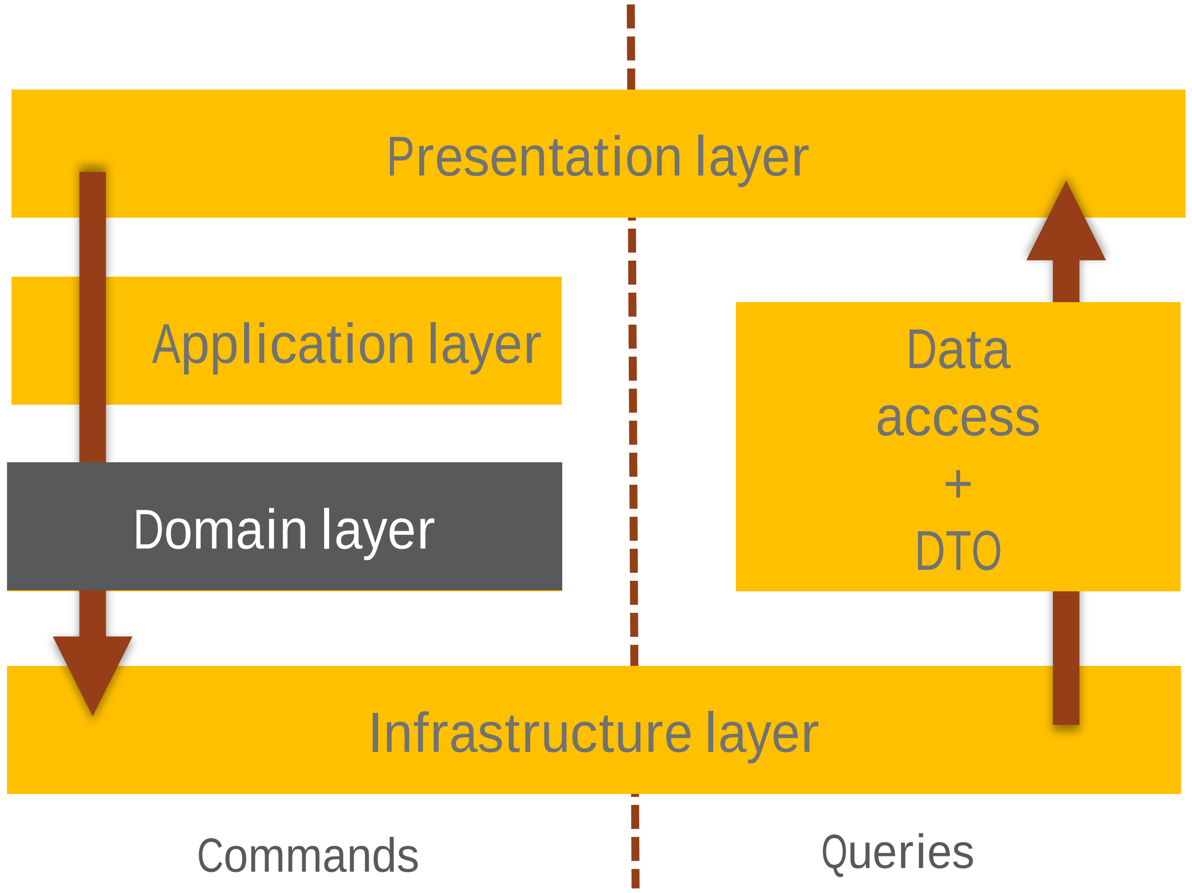


Bertrand Meyer called it
Command/Query Separation back in the 1980s

Canonical **Layered Architecture**

CQRS





Aspects of **CQRS**

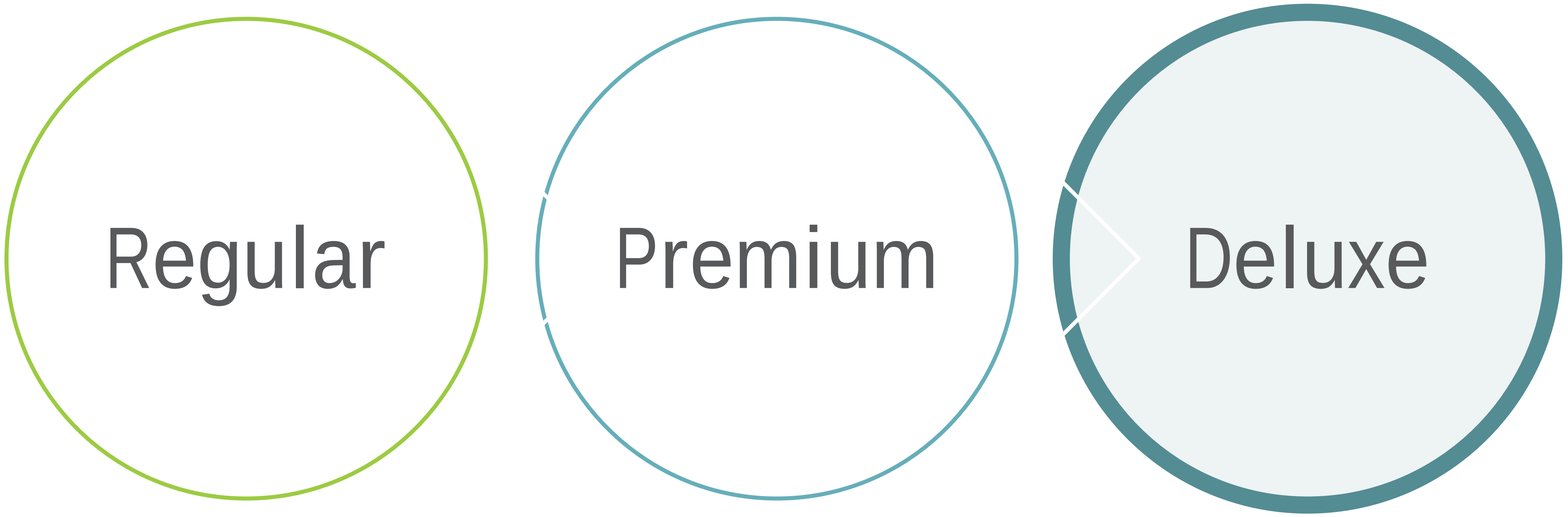
Benefits

- Distinct **optimization**
- **Scalability** potential

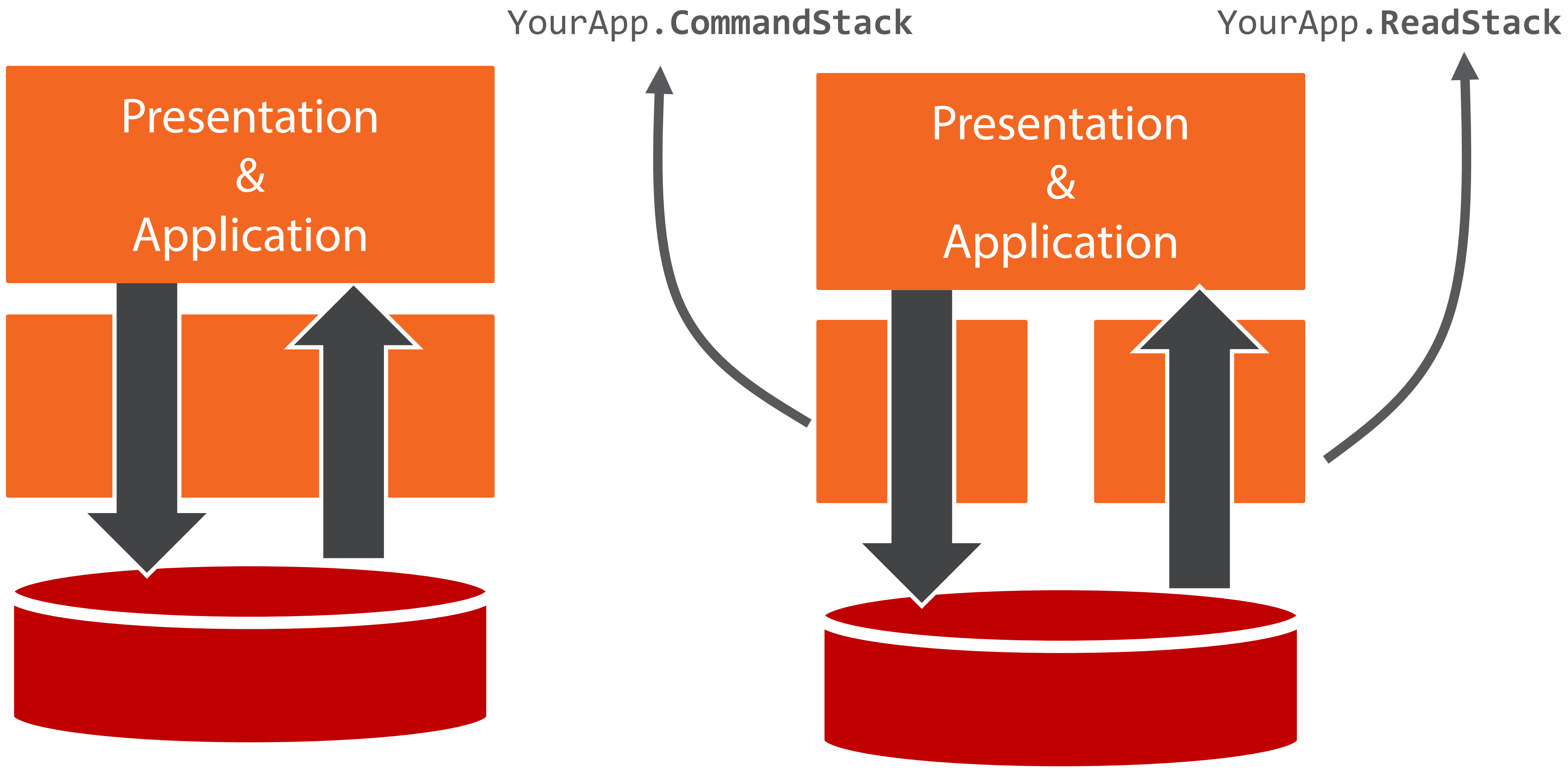
Side effects

- **Simplified** design
- Hassle-free stacks **enhancement**

Flavors of CQRS



CQRS for Plain CRUD Applications



Command Stack

Use just the pattern that fits better

- Existing code
- Existing products
- Existing skills

- Domain Model
- Table Module
- Transaction Script

Read Stack

Use just the code that does the job

- O/RM of choice
- LINQ
- Database in use

TIP

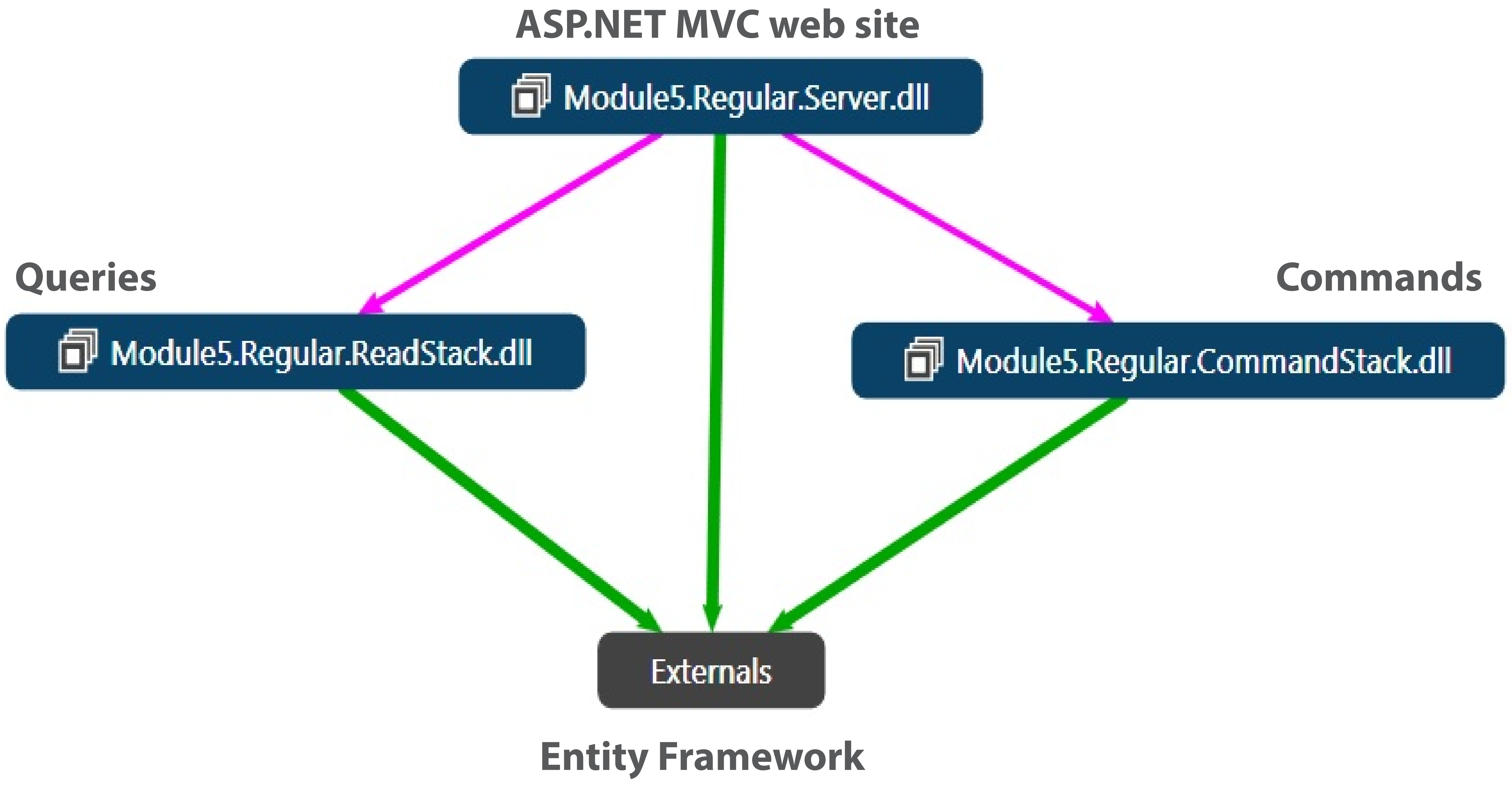
Use a read-only wrapper for the **DbContext** instance you use in the read stack.

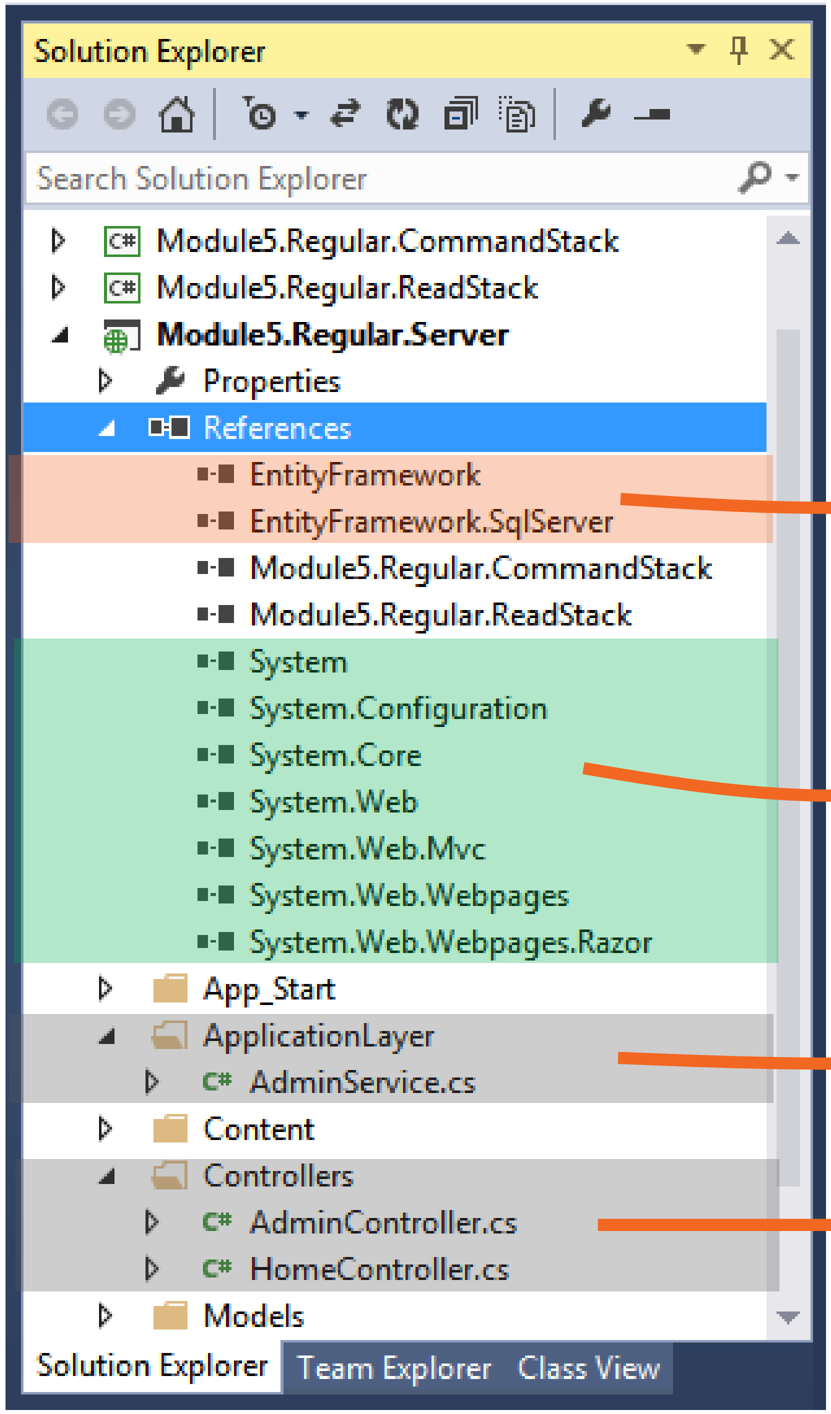
Read-only Database Facade

```
public class Database : IDisposable
{
    private readonly QueryDbContext _db = new QueryDbContext();
    public IQueryable<Customer> Customers
    {
        get { return _db.Customers; }
    }
    public void Dispose()
    {
        _db.Dispose();
    }
}
```

DEMO

CQRS Regular
in action


ASP.NET MVC web site
Module5.Regular.Server.dll
Queries
Module5.Regular.ReadStack.dll
Commands
Module5.Regular.CommandStack.dll
Externals
Entity Framework


Solution Explorer

Search Solution Explorer

- Module5.Regular.CommandStack
- Module5.Regular.ReadStack
- **Module5.Regular.Server**
  - Properties
  - References
    - EntityFramework
    - EntityFramework.SqlServer
    - Module5.Regular.CommandStack
    - Module5.Regular.ReadStack
    - System
    - System.Configuration
    - System.Core
    - System.Web
    - System.Web.Mvc
    - System.Web.Webpages
    - System.Web.Webpages.Razor
  - App_Start
  - ApplicationLayer
    - AdminService.cs
  - Content
  - Controllers
    - AdminController.cs
    - HomeController.cs
  - Models

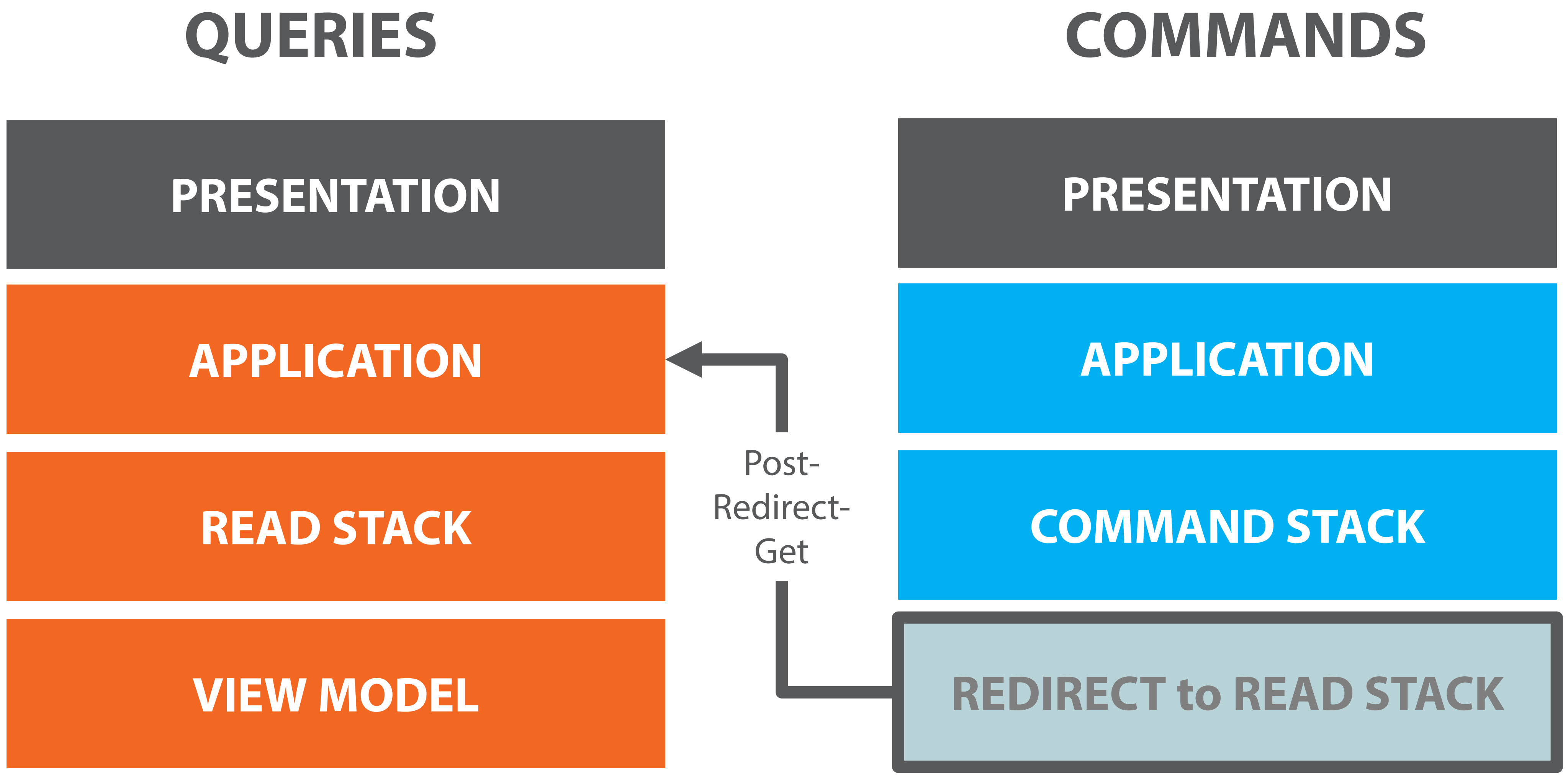
Solution Explorer | Team Explorer | Class View

Externals

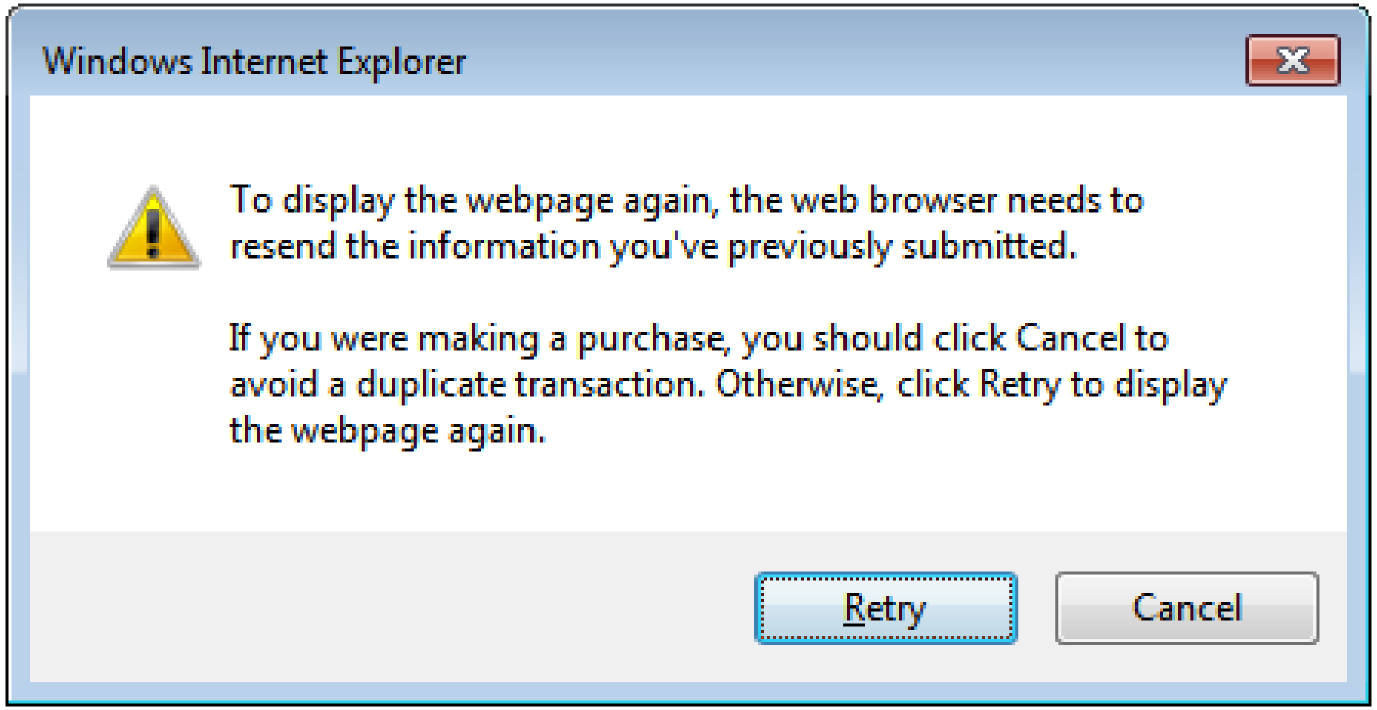
ASP.NET

Application Layer

Presentation Layer


QUERIES
COMMANDS
PRESENTATION
PRESENTATION
APPLICATION
APPLICATION
READ STACK
COMMAND STACK
VIEW MODEL
REDIRECT to READ STACK
Post-Redirect-Get

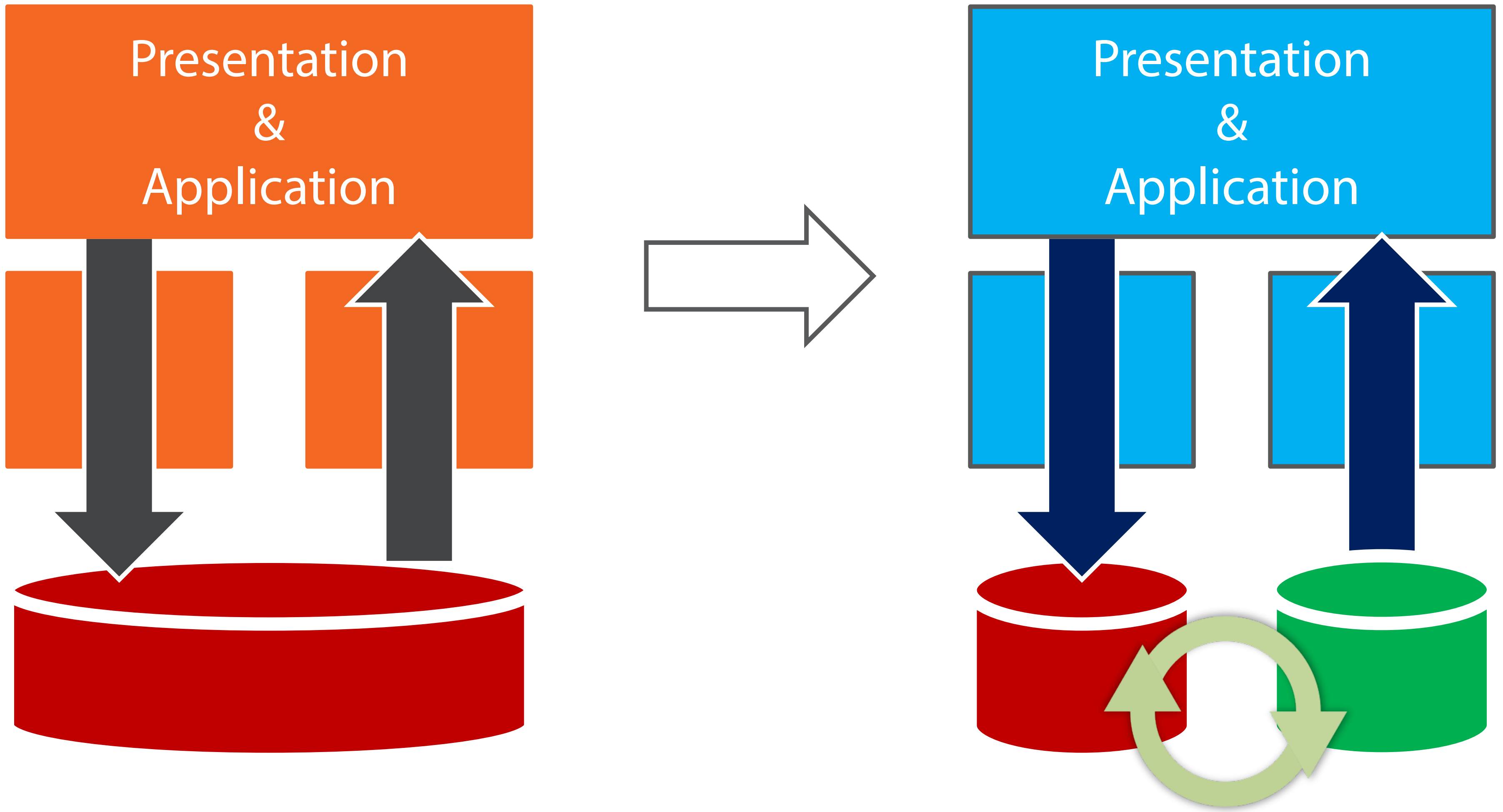
CQRS and **Post-Redirect-Get** web pattern



To display the webpage again, the web browser needs to resend the information you've previously submitted.

If you were making a purchase, you should click Cancel to avoid a duplicate transaction. Otherwise, click Retry to display the webpage again.

CQRS Premium



Command Stack

Use just the pattern that fits better

Task-oriented

Ad-hoc storage

Relational

NoSQL

Events

Domain Model

Transaction Script

Read Stack

Use just the code that does the job

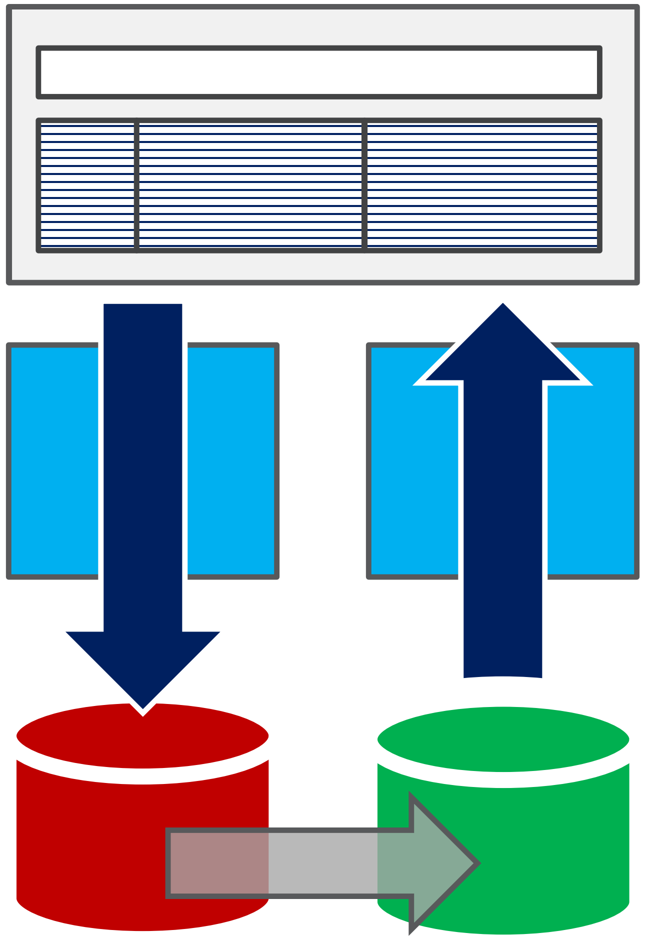
- O/RM of choice
- LINQ
- Ad-hoc storage

Relational

ISSUE

How would you define stale data? How critical is stale data for the app?

Command & Query **Storage**



Command & Query **Storage Synchronization**

Automatically up-to-date

Synchronous

Every command triggers sync updates

Eventually up-to-date

Asynchronous

Every command triggers async updates

Controlled staleness

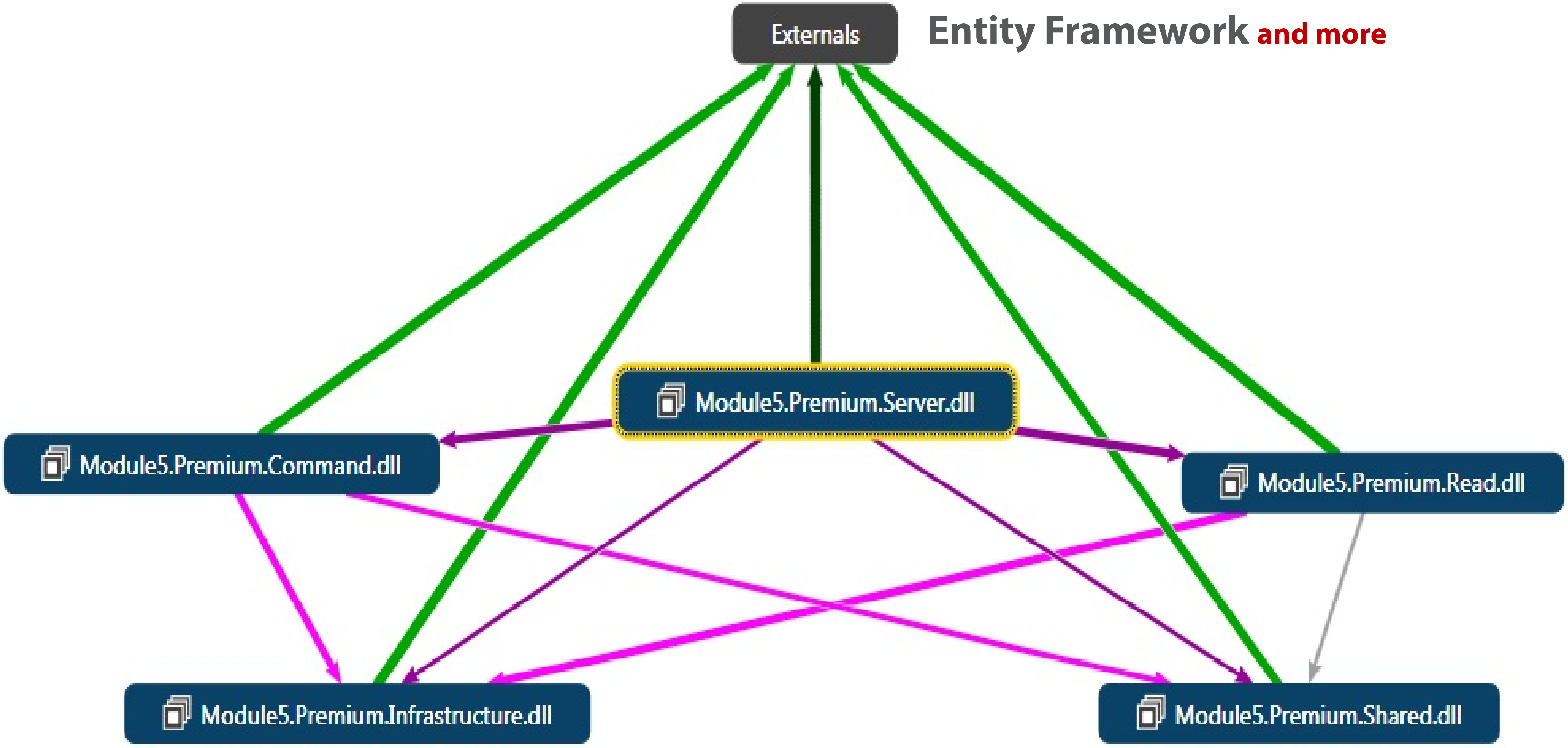
Scheduled

A job runs periodically and updates the read storage

Controlled up-to-date

On-demand

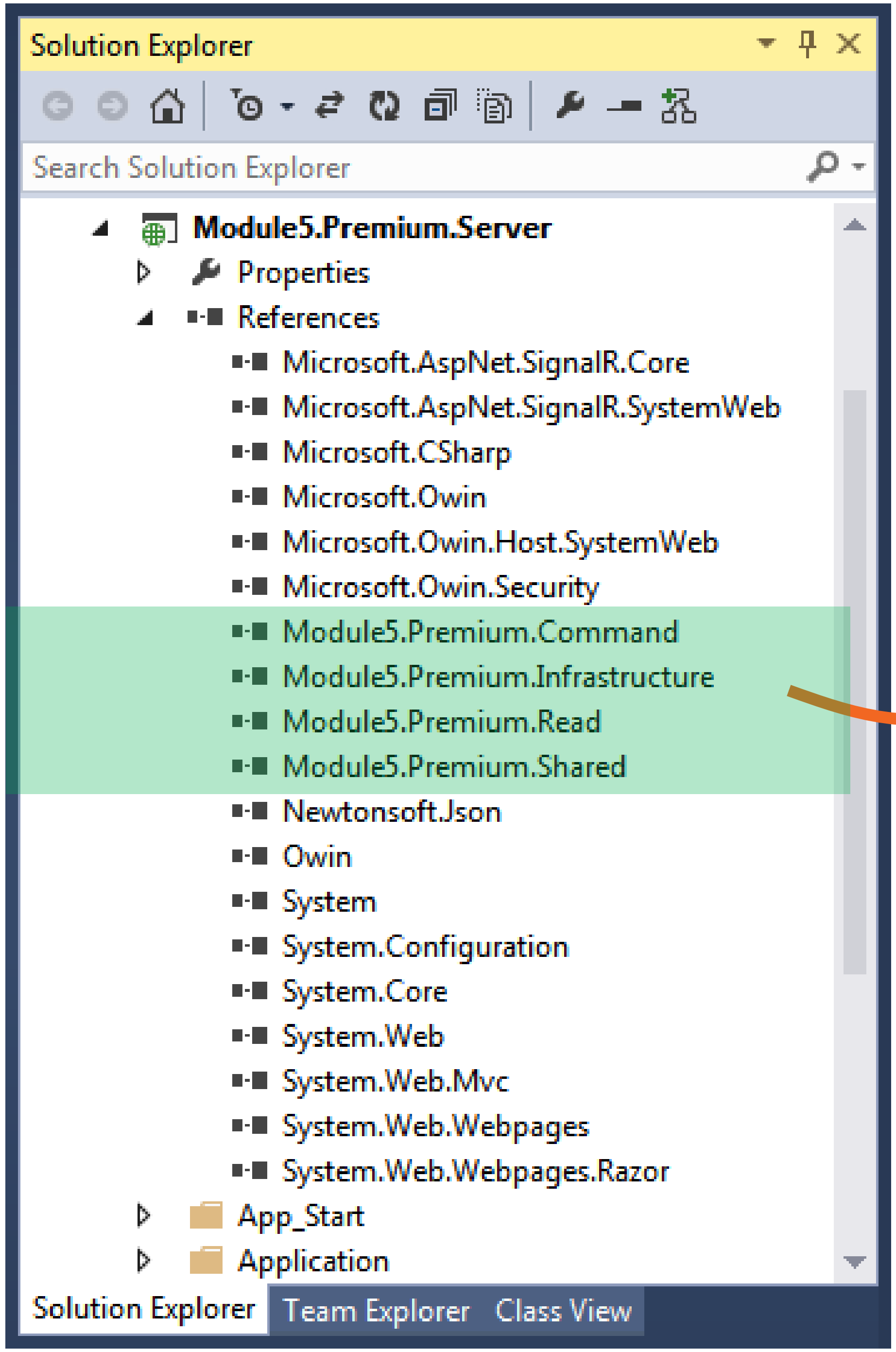
Updates triggered by requests (if older enough)


Externals
Module5.Premium.Server.dll
Module5.Premium.Command.dll
Module5.Premium.Read.dll
Module5.Premium.Infrastructure.dll
Module5.Premium.Shared.dll

Entity Framework and more

Shared data access layer

Helper classes


Solution Explorer

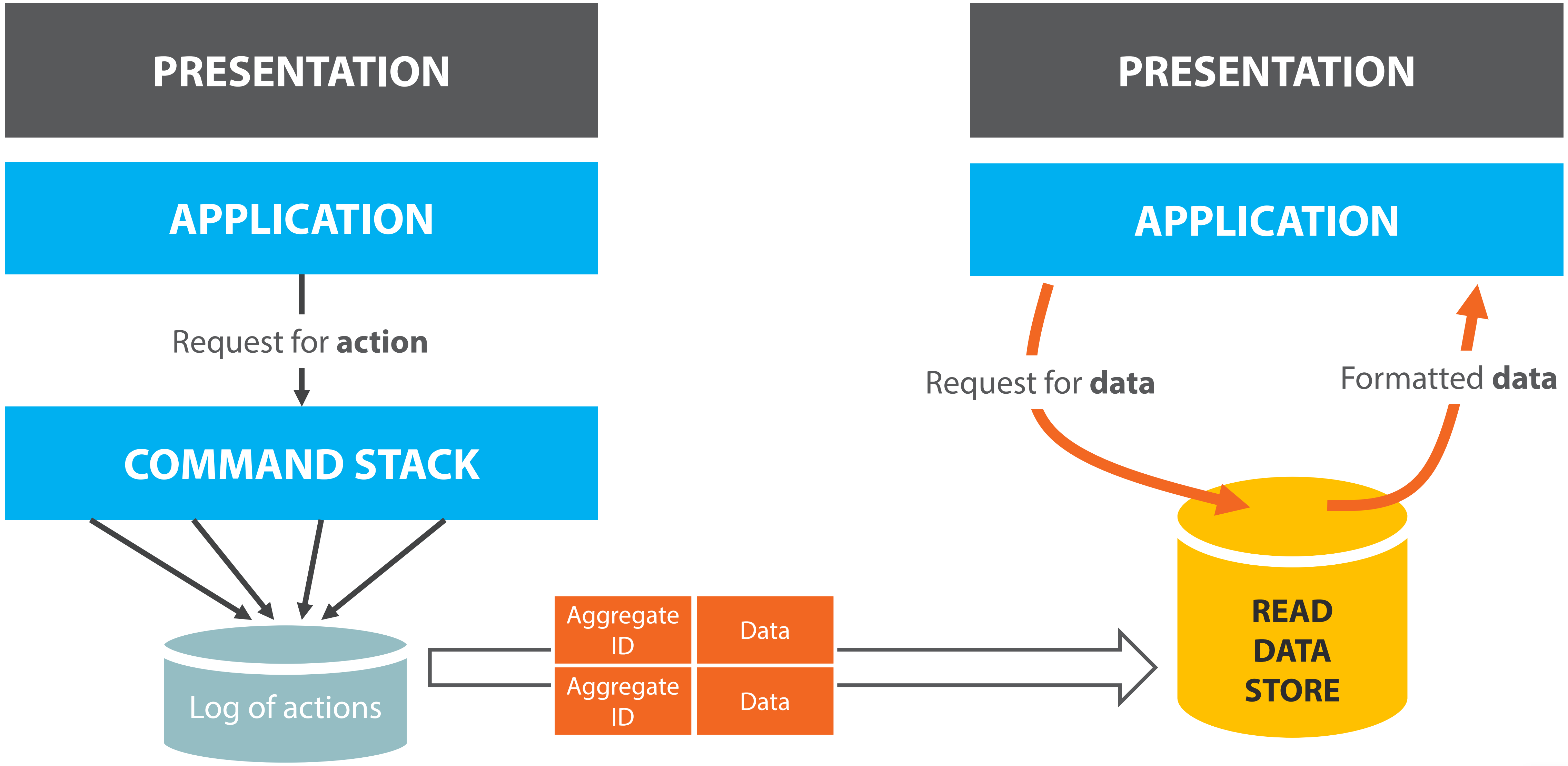
Search Solution Explorer

```
Module5.Premium.Server
  Properties
  References
    Microsoft.AspNet.SignalR.Core
    Microsoft.AspNet.SignalR.SystemWeb
    Microsoft.CSharp
    Microsoft.Owin
    Microsoft.Owin.Host.SystemWeb
    Microsoft.Owin.Security
    Module5.Premium.Command
    Module5.Premium.Infrastructure
    Module5.Premium.Read
    Module5.Premium.Shared
    Newtonsoft.Json
    Owin
    System
    System.Configuration
    System.Core
    System.Web
    System.Web.Mvc
    System.Web.Webpages
    System.Web.Webpages.Razor
  App_Start
  Application
```

Solution Explorer | Team Explorer | Class View

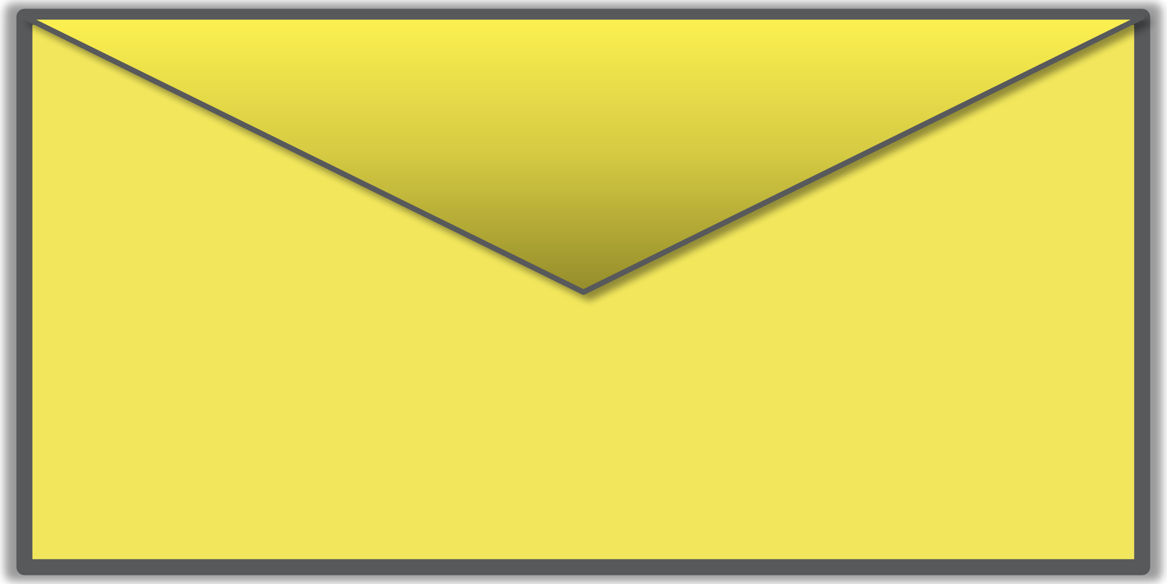
Command stack
Read stack
Infrastructure layer

SYNCHRONIZATION



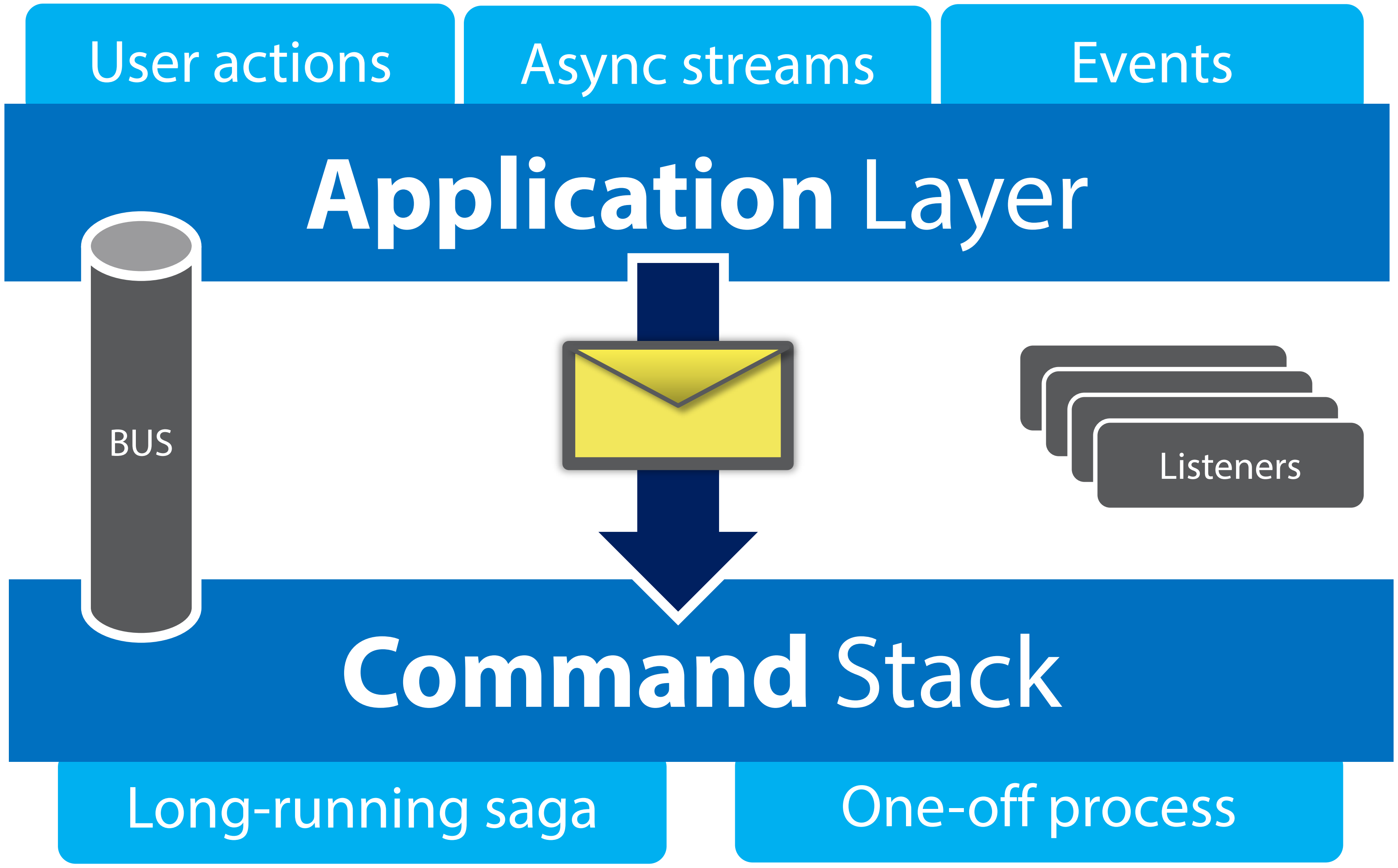
DEMO

Power to Messages

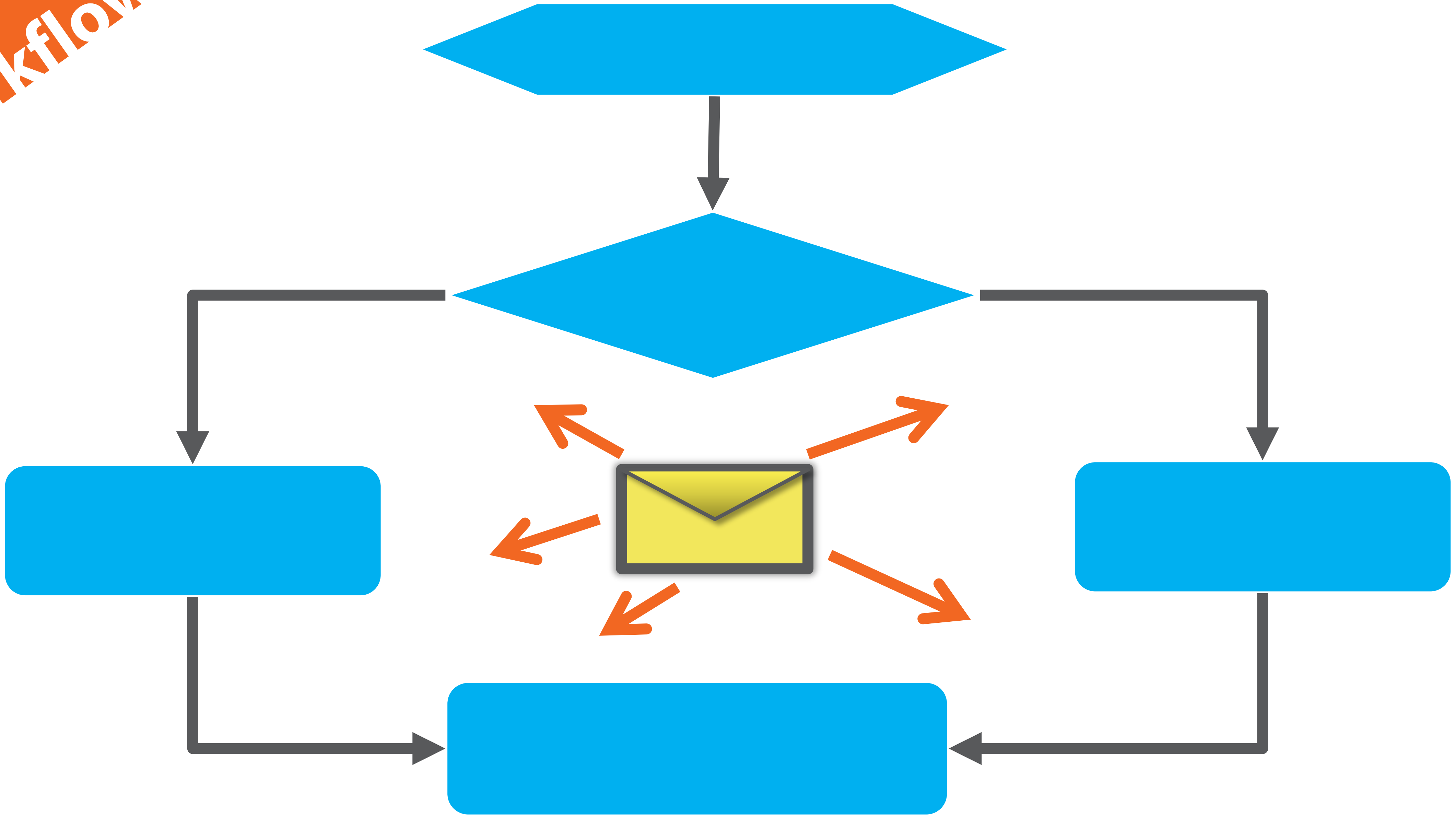


Representing Messages

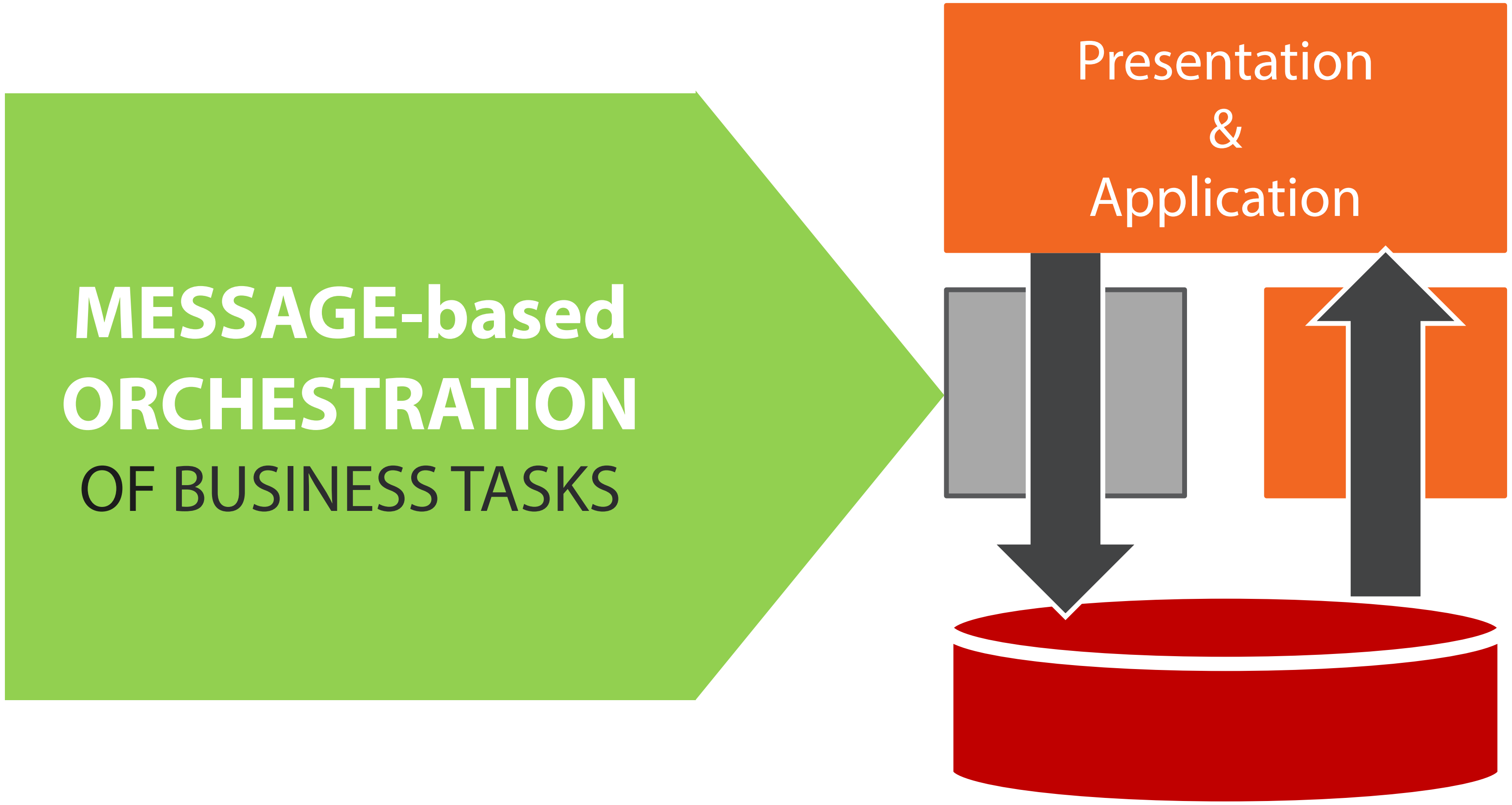
```
public class Message
{
   public DateTime TimeStamp { get; protected set; }
   public String SagaId { get; protected set; }
}

public class Command : Message
{
   public String Name { get; protected set; }
}

public class Event : Message
{
   // Any properties that may help retrieving and persisting events.
}
```


User actions
Async streams
Events
Application Layer
BUS
Listeners
Command Stack
Long-running saga
One-off process

Workflow

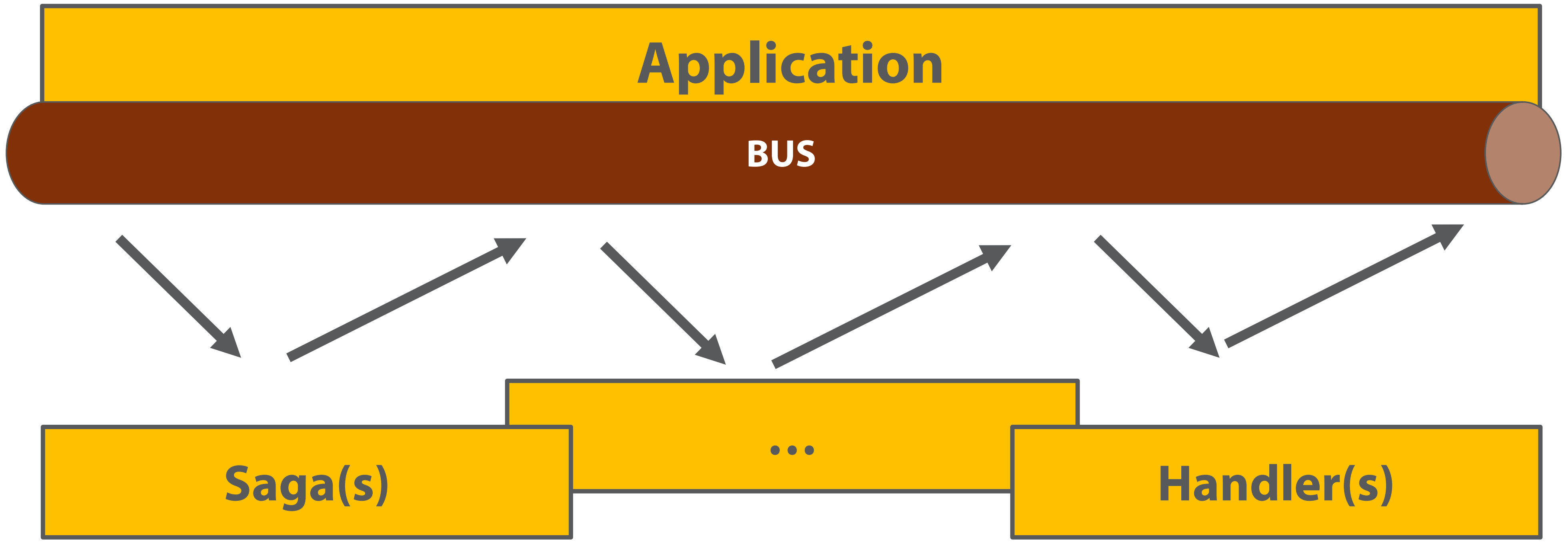


CQRS **Deluxe**


MESSAGE-based
ORCHESTRATION
OF BUSINESS TASKS
Presentation
&
Application

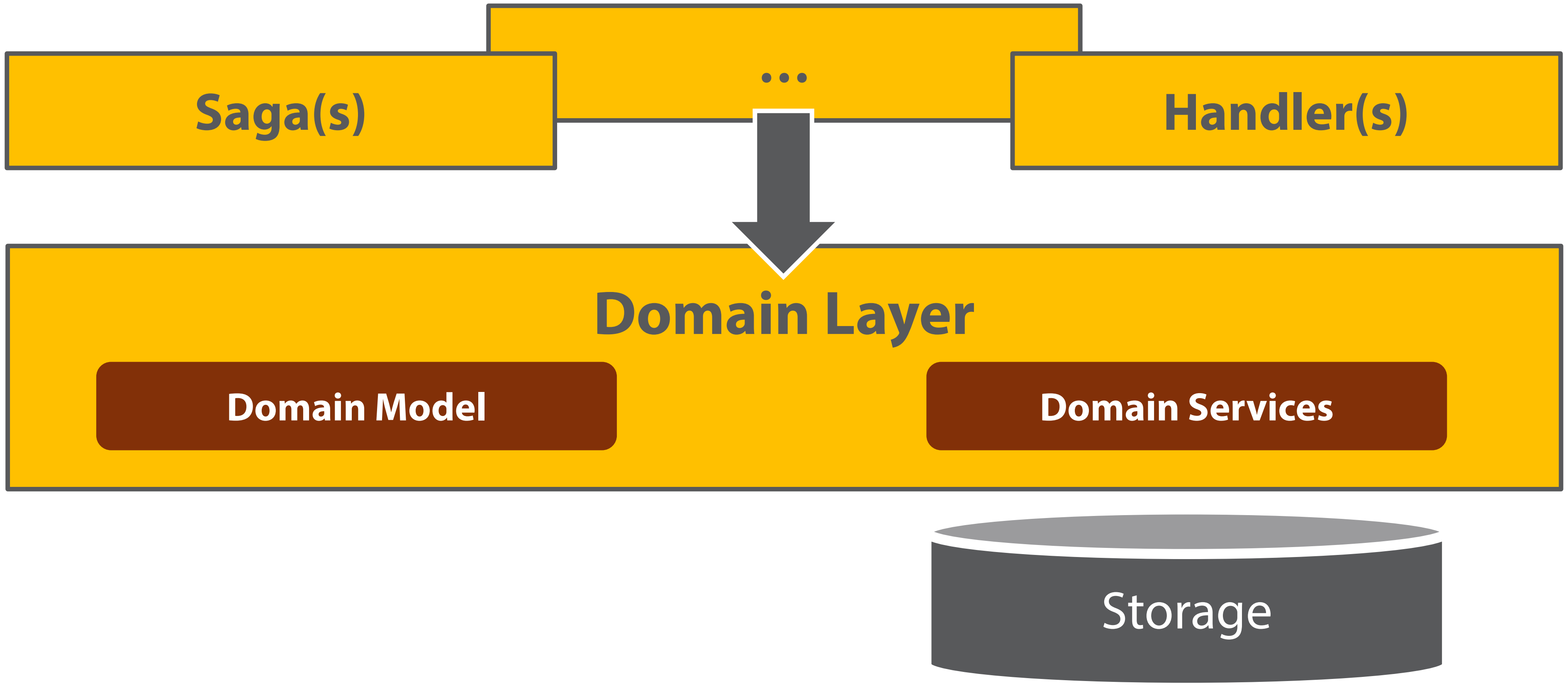
Command Stack

Use just the pattern that fits better

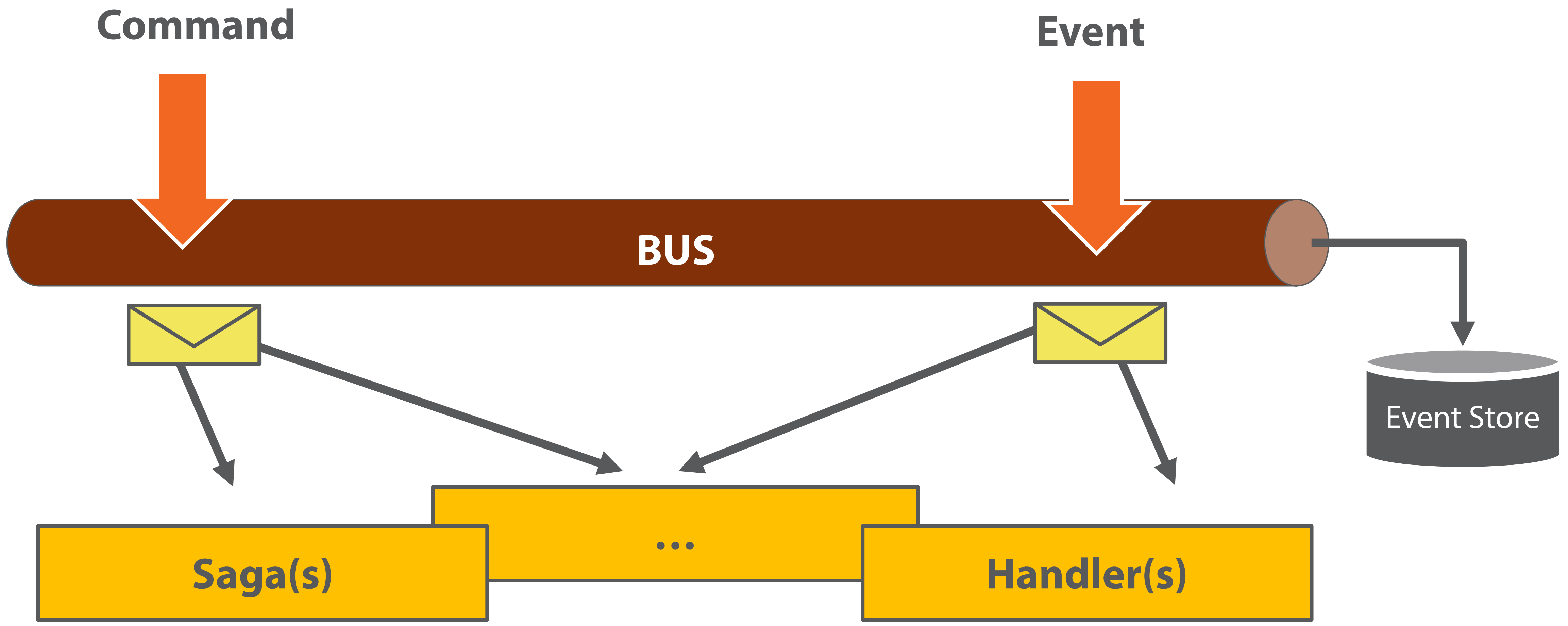


Command Stack

Use just the pattern that fits better



Storage



Read Stack

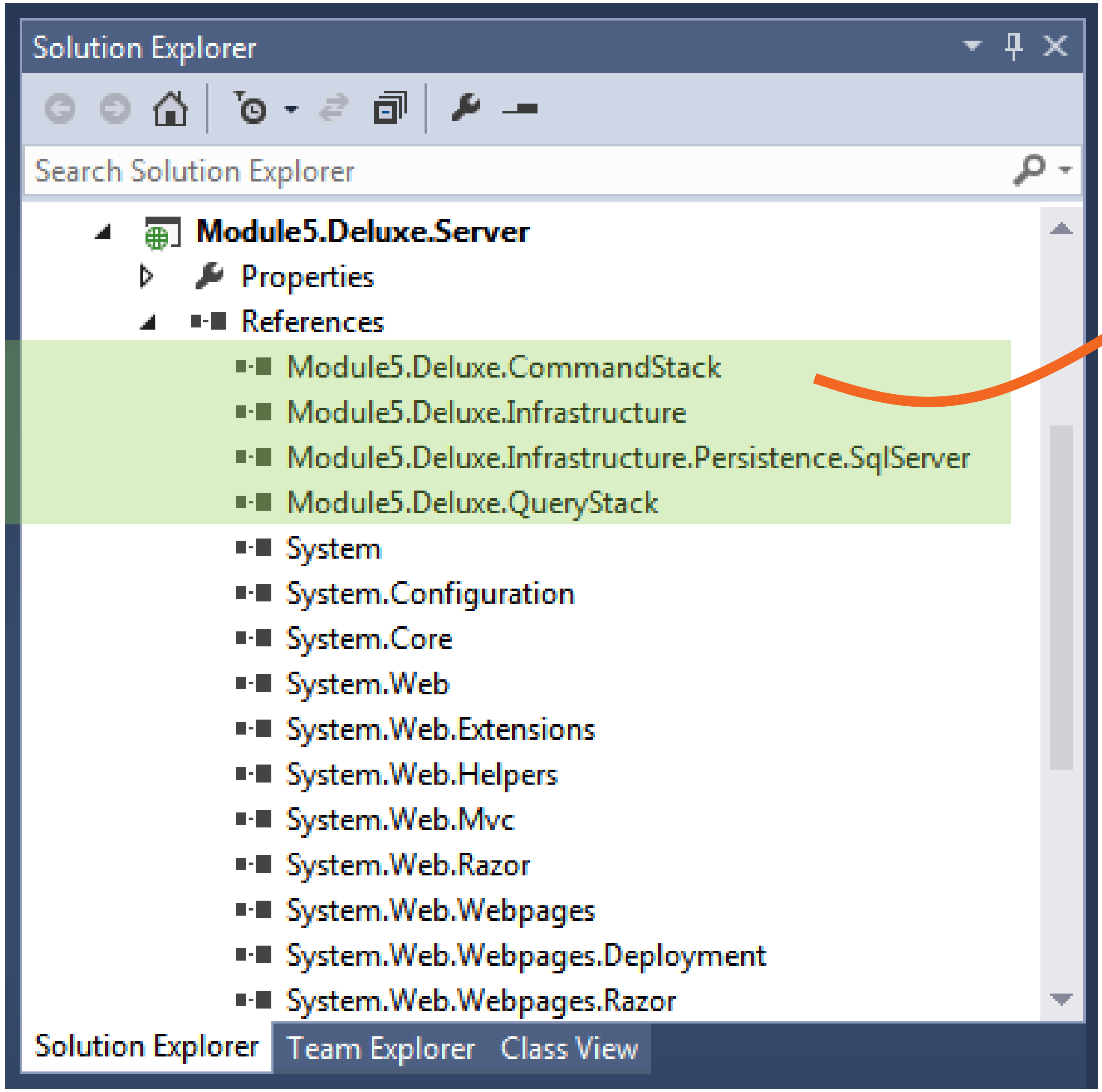
Use just the code that does the job

- O/RM of choice
- LINQ
- Ad-hoc storage

Relational

How would you build the read snapshot database?

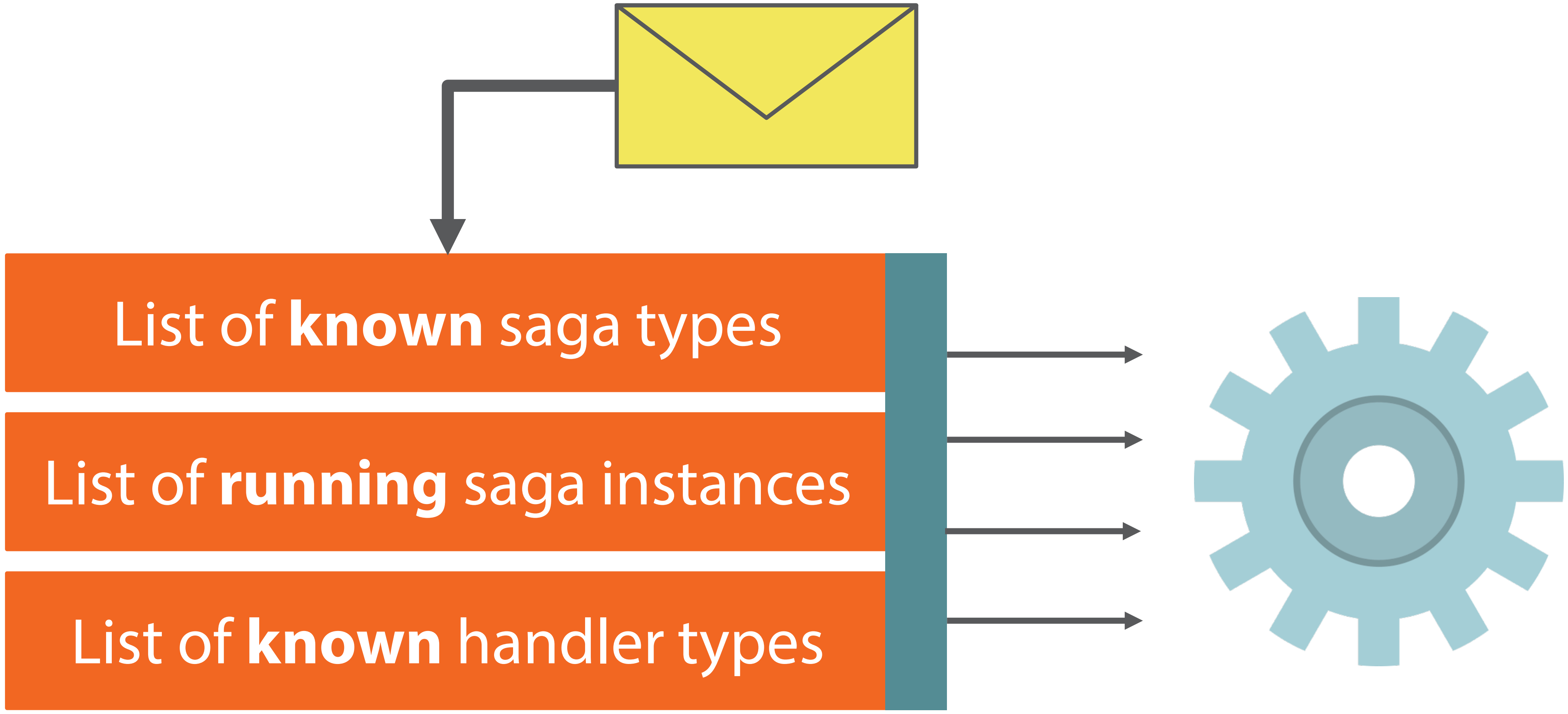
Use a handler!



- Module5.Deluxe.CommandStack
- Module5.Deluxe.Infrastructure
- Module5.Deluxe.Infrastructure.Persistence.SqlServer
- Module5.Deluxe.QueryStack
- System
- System.Configuration
- System.Core
- System.Web
- System.Web.Extensions
- System.Web.Helpers
- System.Web.Mvc
- System.Web.Razor
- System.Web.Webpages
- System.Web.Webpages.Deployment
- System.Web.Webpages.Razor

- **Command stack**
- **Read stack**
- **Infrastructure layer**
  (with persistence)

INSIDE THE BUS


List of **known** saga types
List of **running** saga instances
List of **known** handler types

INSIDE THE SAGA

Command or event that starts the associated business process

List of commands the saga can handle

List of events the saga is interested in

```
public class CheckoutSaga : Saga<CheckoutSagaData>,
        IStartWith<StartCheckoutCommand>,
        ICanHandle<CancelCheckoutCommand>,
        ICanHandle<PaymentCompletedEvent>,
        ICanHandle<PaymentDeniedEvent>,
        ICanHandle<DeliveryRequestRefusedEvent>,
        ICanHandle<DeliveryRequestApprovedEvent>
{
    public void Handle(StartCheckoutCommand message)
    {
        :
    }

    ...
}
```

More About Sagas

Sagas must be identified by a unique ID	• Can be a GUID • Can be the ID of the aggregate the saga is all about • Can be a combination of values that is unique in the context
Sagas might be persistent and stateful	• Persistence is care of the bus • State of the associated aggregate must be persisted
Sagas might be stateless	• Mere task executor getting any data from the start command

Extending a Solution

Got a new handling scenario for an existing event?

Got a new handling scenario for a new feature?

Write a new saga or handler and **register** it with the bus.
That’s it.

More About the Bus

- **Can write your own bus**
  - Mostly about real traffic hitting the application
  - Plug in some queue engine
  - Plug in some persistence mechanism

- **Look into existing products and frameworks**
  - **NServiceBus** from Particular Software: particular.net/nservicebus
  - **Rebus** from Rebus-org: github.com/rebus-org/Rebus
  - **MassTransit** from Pandora: masstransit-project.com

DEMO