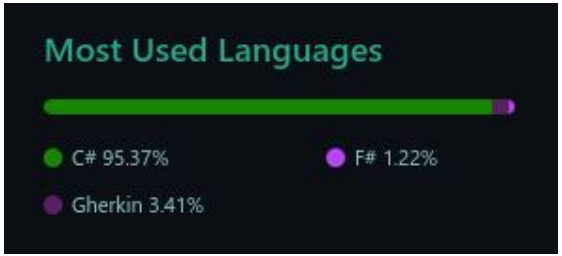
VONAGE

# Throw exceptions... Out of your codebase

PRESENTED BY:
Guillaume Faas
.Net Developer Advocate,
Vonage

# About me

- Software Craftsman
- .Net Developer Advocate @ Vonage
- Coach & Speaker @ Les Tontons Crafters (for fun)
- Meetup Host @ Software Craftsmanship Luxembourg




Tr00d

guillaumefaas

# Why do I want to talk about Exceptions?

throw new MissingContextException(“People need to know!”);

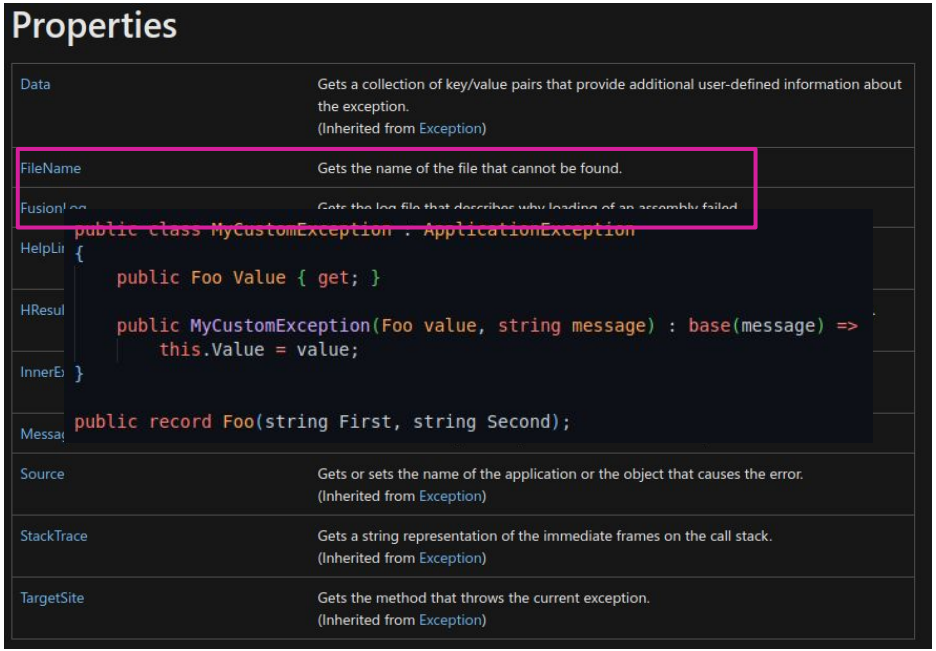
“ exception handling [...] help you deal with any **unexpected** or **exceptional** situations [...] when a program is running “

(https://learn.microsoft.com/en-us/dotnet/csharp/fundamentals/exceptions/)

```
public class ExceptionTest
{
    static double SafeDivision(double x, double y)
    {
        if (y == 0)
            throw new DivideByZeroException();
        return x / y;
    }

    public static void Main()
    {
        // Input for test purposes. Change the values to see
        // exception handling behavior.
        double a = 98, b = 0;
        double result;

        try
        {
            result = SafeDivision(a, b);
            Console.WriteLine("{0} divided by {1} = {2}", a, b, result);
        }
        catch (DivideByZeroException)
        {
            Console.WriteLine("Attempted divide by zero.");
        }
    }
}
```

# Pros

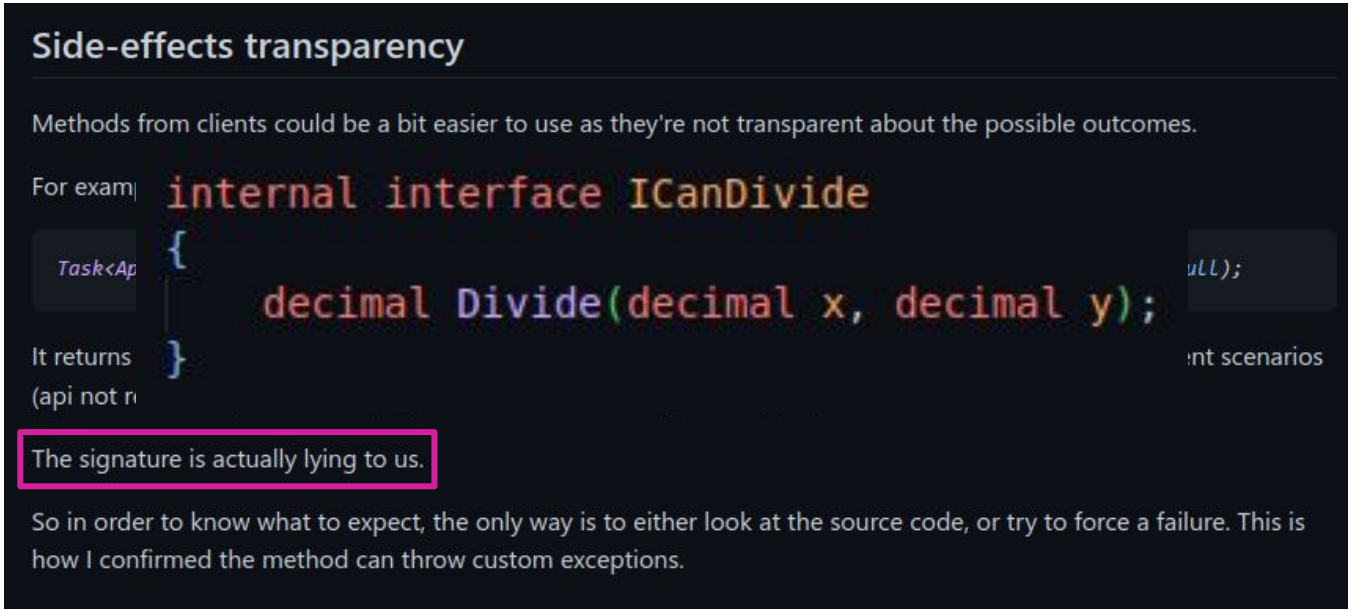
- Allows to “short-circuit” a process from anywhere
- Can carry data
  - Type
  - Content
- Can make your own
- Easy to use
- Exist in most languages and frameworks



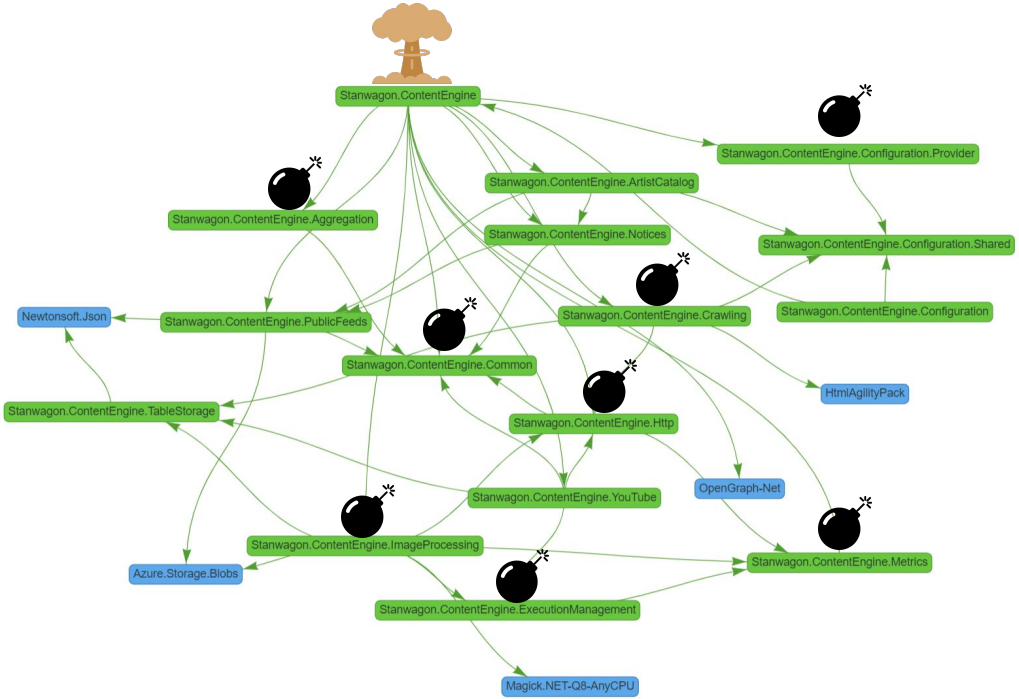
# It’s intrusive

```
try
{
    result = SafeDivision(a, b);
    Console.WriteLine("{0} divided by {1} = {2}", a, b, result);
}
catch (DivideByZeroException)
{
    Console.WriteLine("Attempted divide by zero.");
}
```

# Unpredictable from the caller

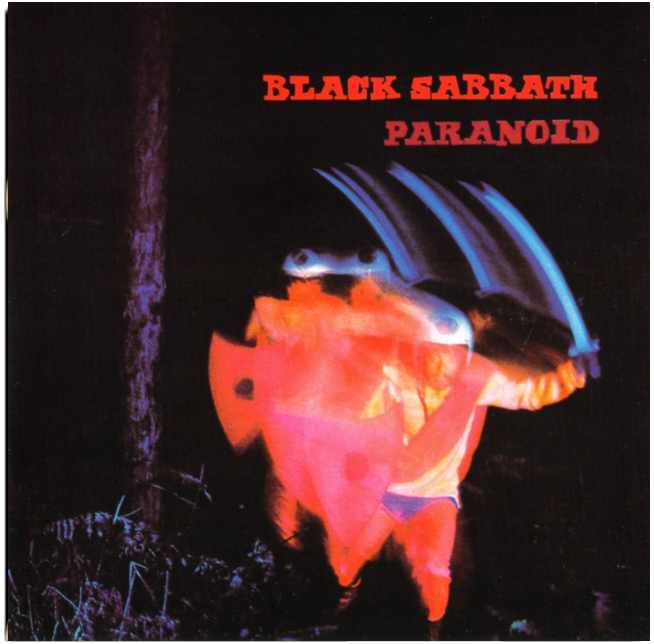


# Makes flow hard to understand



# What I observed

- Exceptions are over-used
- Exception paranoia
  - *“Will this method throw an exception?”*



# Is there an alternative?

# Requirements?

- Easy way to expose a failure
- Transparency / Predictability
- Flow easy to read

# “We should not speak its name”


WANTED
DEAD & ALIVE
SCHRÖDINGER'S CAT

# A cat in a box

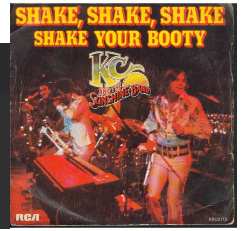
```
internal class SchrodingerBox
{
    private readonly Cat? cat;

    // Creates a box with an alive cat inside
    private SchrodingerBox(Cat cat) => this.cat = cat;

    // Creates a box with a dead cat
    private SchrodingerBox()
    {
    }

    // Look inside the box
    public Cat? OpenBox() => this.cat;

    // Shakes the box. The cat doesn't like it. Meow
    public SchrodingerBox Shake()
    {
        this.cat?.Meow();
        return this;
    }

    // Shakes the box (too hard), then returns a new box with a dead cat
    public SchrodingerBox ShakeTooHard()
    {
        this.Shake();
        return new SchrodingerBox();
    }

    // Creates a box with a cat
    public static SchrodingerBox WithAliveCat(Cat cat) => new(cat);
}
```

** No cat was harmed during the preparation of this talk*

# Shake the box - Don’t shake the cat

```
[Fact]
public void AliveCatMeowsMeowsMeows() =>
    SchrodingerBox.WithAliveCat(this.GetAliveCat())
        .Shake()
        .Shake()
        .Shake();
```

```
AliveCatMeowsMeowsMeows [10 ms]
Meow
Meow
Meow
```




```
[Fact]
public void ShakeTooHardUnfortunatelyKillsTheCat() =>
    SchrodingerBox.WithAliveCat(this.GetAliveCat())
        .Shake()
        .ShakeTooHard()
        .Shake();
```

```
ShakeTooHardUnfortunatelyKillsTheCat [0 ms]
Meow
```

```
string actual = "ABCDEFGHI";

actual.Should().StartWith("AB").And.EndWith("HI").And.Contain("EF").And.HaveLength(9);
```

```
[Fact]
public void OpenBox_ReturnsDeadCat_WhenCatIsDead() =>
    SchrodingerBox.WithAliveCat(this.GetAliveCat())
        .ShakeTooHard()
        .OpenBox()
        .Should()
        .BeNull();
```

```
OpenBox_ReturnsDeadCat_WhenCatIsDead [0 ms]
Meow
```

```
[Fact]
public void OpenBox_ReturnsAliveCat_WhenCatIsAlive() =>
    SchrodingerBox.WithAliveCat(this.GetAliveCat())
        .Shake()
        .OpenBox()
        .Should()
        .Be(this.GetAliveCat());
```

```
OpenBox_ReturnsAliveCat_WhenCatIsAlive [12 ms]
Meow
```

# A <T> in a box

Null


Instance

# No more shaking!

```
internal class SchrodingerBox<T> where T : struct
{
    private bool IsSome { get; }
    private readonly T value;

    // Constructor for Some
    private SchrodingerBox(T value)
    {
        this.value = value;
        this.IsSome = true;
    }

    // Constructor for None
    private SchrodingerBox() => this.IsSome = false;

    // Applies the function on the value IF our box contains a value
    public SchrodingerBox<T> Map(Func<T, T> map) =>
        this.IsSome
            ? Some(map(this.value))
            : None();

    // Creates a box without a value
    public static SchrodingerBox<T> None() => new();

    public T? OpenBox() => this.IsSome ? this.value : null;

    // Creates a box with a value
    public static SchrodingerBox<T> Some(T value) => new(value);
}
```

```
int? value = 3;
if (value != null)
{
    value += 1;
}

if (value != null)
{
    value += 1;
}

if (value != null)
{
    value += 1;
}

var result = value != null ? value : null;
result.Should().Be(6);
```

Source: Adit Bhargava - Functors, applicatives and monads in pictures
https://www.adit.io/posts/2013-04-17-functors,_applicatives,_and_monads_in_pictures.html#just-what-is-a-functor,-really

# Avoid null when opening the box

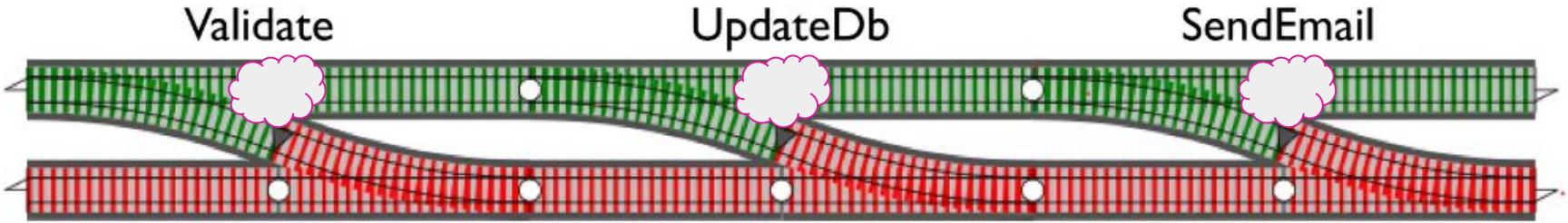
```
[Fact]
public void Match_ShouldExecuteSomeFunction_GivenValueIsSome() =>
    SchrodingerBox<int>.Some(3)
        .Map(value => value + 1)
        .Map(value => value + 1)
        .Map(value => value + 1)
        .Match(some => $"The value is some {some}!", () => "The value is none")
        .Should()
        .Be("The value is some 6!");

[Fact]
public void Match_ShouldExecuteNoneFunction_GivenValueIsNone() =>
    SchrodingerBox<int>.None()
        .Map(value => value + 1)
        .Map(value => value + 1)
        .Map(value => value + 1)
        .Match(some => $"The value is some {some}!", () => "The value is none")
        .Should()
        .Be("The value is none");
```

# Declarative writing

```
[Fact]
public void Match_ShouldExecuteSomeFunction_Declarative() =>
    SchrodingerBox<int>.Some(3)
        .Map(IncrementValue)
        .Map(IncrementValue)
        .Map(IncrementValue)
        .Match(ConvertToMessage, GetDefaultMessage)
        .Should()
        .Be(ConvertToMessage(6));
```

```
[Fact]
public void Match_ShouldExecuteNoneFunction_Declarative() =>
    SchrodingerBox<int>.None()
        .Map(IncrementValue)
        .Map(IncrementValue)
        .Map(IncrementValue)
        .Match(ConvertToMessage, GetDefaultMessage)
        .Should()
        .Be(GetDefaultMessage());
```

# Railway-oriented programming




Source: "*Railway Oriented Programming Error handling in functional languages*" by Scott Wlaschin

https://swlaschin.gitbooks.io/fsharpforfunandprofit/content/posts/recipe-part2.html


“Som tand,

# Which monads?

## LanguageExt

https://github.com/louthy/language-ext

- Monads
  - **Option/Maybe** - Represents an optional value (Some/None)
  - **Try** - Represents a process that can result in throwing an exception
  - **Either** - Represents a value of two possible types
- Immutable collections
- “Make C# better”

# Language-Ext version

```
private static Either<Error, decimal> Divide(decimal value, decimal divisor) =>
    divisor == 0
        ? new Error("Cannot divide by 0.")
        : value / divisor;

private record Error(string Message);
```

```
[Fact]
public void Divide_ShouldReturnLeft_GivenDivisorIsZero() =>
    Divide(50, 0)
        .Should()
        .BeLeft(left => left.Should().Be(new Error("Cannot divide by 0.")));

[Fact]
public void Divide_ShouldReturnRight_GivenValueCanBeDivided() =>
    Divide(50, 2)
        .Should()
        .BeRight(value => value.Should().Be(25));
```

# Language-Ext version

```
public class ExceptionTest
{
    static double SafeDivision(double x, double y)
    {
        if (y == 0)
            throw new DivideByZeroException();
        return x / y;
    }

    public static void Main()
    {
        // Input for test purposes. Change the values to see
        // exception handling behavior.
        double a = 98, b = 0;
        double result;

        try
        {
            result = SafeDivision(a, b);
            Console.WriteLine("{0} divided by {1} = {2}", a, b, result);
        }
        catch (DivideByZeroException)
        {
            Console.WriteLine("Attempted divide by zero.");
        }
    }
}
```

9 LoC

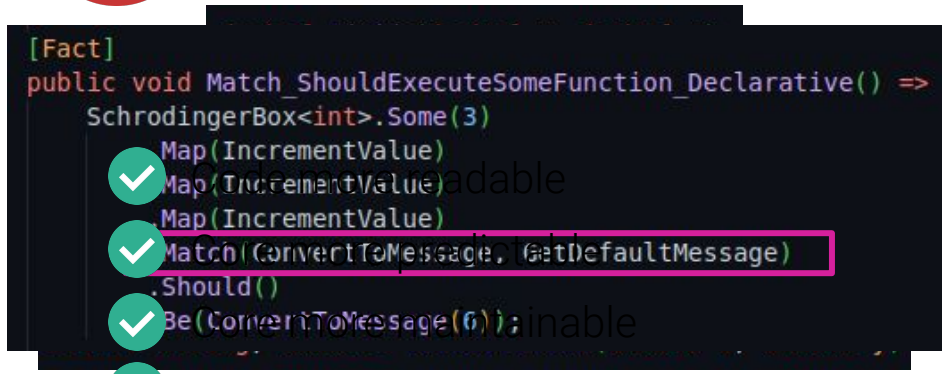
```
public class DivideExampleTest
{
    public void Main()
    {
        double a = 98, b = 0;
        var text = SafeDivision(a, b).Match(success => $"{a} divided by {b} = {success}", _ => _);
        Console.WriteLine(text);
    }

    private static Either<string, double> SafeDivision(double x, double y) =>
        y == 0
            ? Either<string, double>.Left("Attempted divide by zero.")
            : Either<string, double>.Right(x / y);
}
```

**2** LoC

# Monads are awesome

- Referential transparency
- Reduces branching
- Encourages a declarative writing
- Forces you to deal all possible states
- Tends to produce smaller codebases

# Reasons you want to use Monads




- Reduce cognitive load

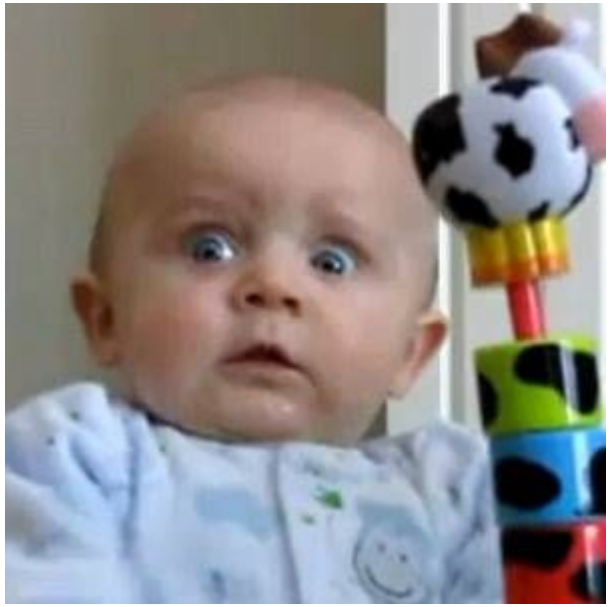
# We need to talk about Functional Programming

No, we won't talk about math...

# Plenty of scary terms...

- Functors
- Monads
- Monoids
- Currying
- Higher-order functions
- Pure functions
- Referential transparency
- etc.

# ... that you're familiar with



# LINQ is functional

```
[Fact]
◇  tr00d *
public void GetTop3Players()
{
    var output = new string[3];
    foreach (var player :Person in this.top10Players)
    {
        if (player.Rank <= 3)
        {
            output[player.Rank - 1] = player.GetDisplayName();
        }
    }

    output // string[]
        .Should() // StringCollectionAssertions
        .BeEquivalentTo( params expectation: "Karim Benzema", "Sadio Mané", "Kevin De Bruyne");
}
```

```
public LinqExampleTest() =>
    this.top10Players = new List<Player>
    {
        new("Erling", "Haaland", 10),
        new("Kylian", "Mbappé", 6),
        new("Luka", "Modrić", 9),
        new("Kevin", "De Bruyne", 3),
        new("Thibault", "Courtois", 7),
        new("Karim", "Benzema", 1),
        new("Robert", "Lewandowski", 4),
        new("Sadio", "Mané", 2),
        new("Mohamed", "Salah", 5),
        new("Vinicius", "Júnior  ", 8),
    };
```

```
[Fact]
public void GetTop3PlayersWithLinq() =>
    this.top10Players
        .OrderBy(player => player.Rank)
        .Take(3)
        .Select(player => player.GetDisplayName())
        .Should()
        .BeEquivalentTo("Karim Benzema", "Sadio Mané", "Kevin De Bruyne");
```

# Higher-order functions

```
[Fact]
public void GetTop3PlayersWithLinq() =>
    this.top10Players
        .OrderBy(player => player.Rank)
        .Take(3)
        .Select(player => player.GetDisplayName())
        .Should()
        .BeEquivalentTo("Karim Benzema", "Sadio Mané", "Kevin De Bruyne");
```

```
OrderBy<TSource,TKey>(IEnumerable<TSource>,
Func<TSource,TKey>)
```

```
Select<TSource,TResult>(IEnumerable<TSource>,
Func<TSource,TResult>)
```

# Pure function

```
[Fact]
public void GetTop3PlayersWithLinq() =>
    this.top10Players
        .OrderBy(player => player.Rank)
        .Take(3)
        .Select(player => player.GetDisplayName())
        .Should()
        .BeEquivalentTo("Karim Benzema", "Sadio Mané", "Kevin De Bruyne");
```

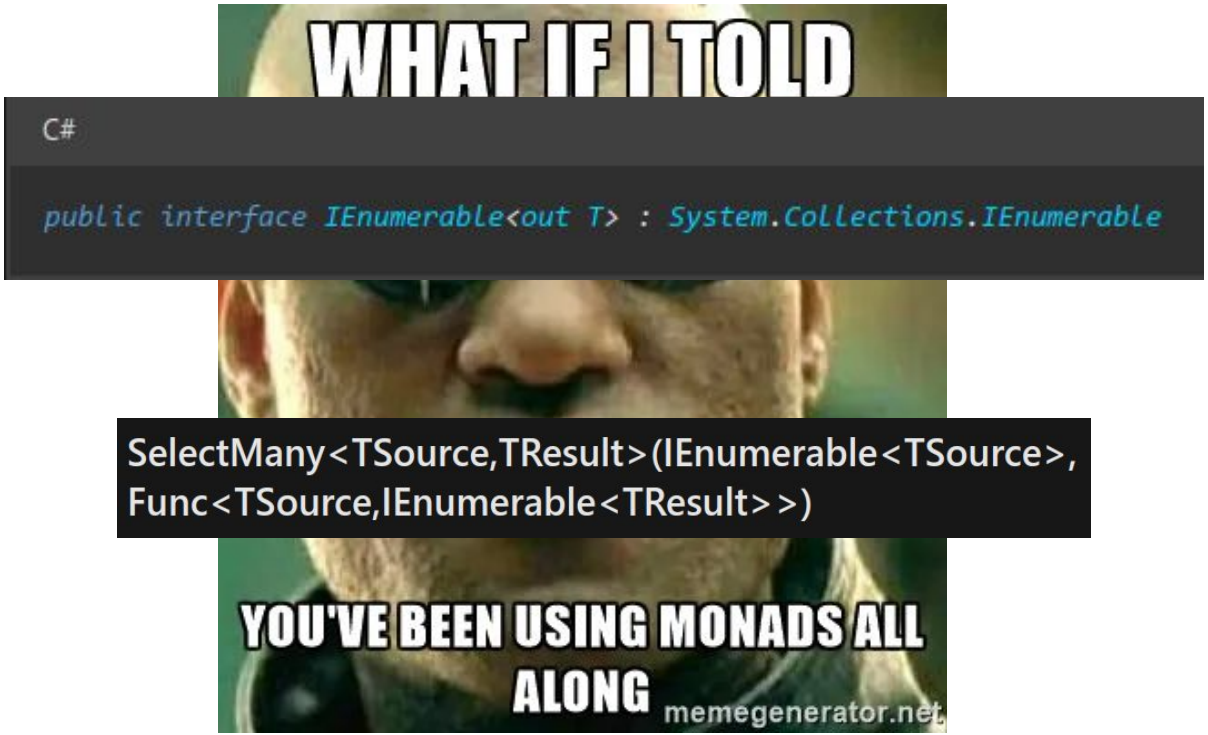
“When a function is pure, we say output depends only on input”
- Joshua Backfield, in his book “Becoming Functional”

# Immutability

```
[Fact]
public void GetTop3PlayersWithLinq() =>
    this.top10Players
        .OrderBy(player => player.Rank)
        .Take(3)
        .Select(player => player.GetDisplayName())
        .Should()
        .BeEquivalentTo("Karim Benzema", "Sadio Mané", "Kevin De Bruyne");
```

```
private record Player(string Firstname, string Lastname, int Rank)
{
    public string GetDisplayName() => string.Join(' ', this.Firstname, this.Lastname);
}
```

# Monads you already use



C#

```csharp
private static string ConvertToMessage(int value) => $"The value is some {value}!";

private static string GetDefaultMessage() => "The value is none";

private static int IncrementValue(int value) => value + 1;
```

```csharp
[Fact]
public void Match_ShouldExecuteSomeFunction_Declarative() =>
    SchrodingerBox<int>.Some(3)
        .Map(IncrementValue)
        .Map(IncrementValue)
        .Map(IncrementValue)
        .Match(ConvertToMessage, GetDefaultMessage)
        .Should()
        .Be(ConvertToMessage(6));
```

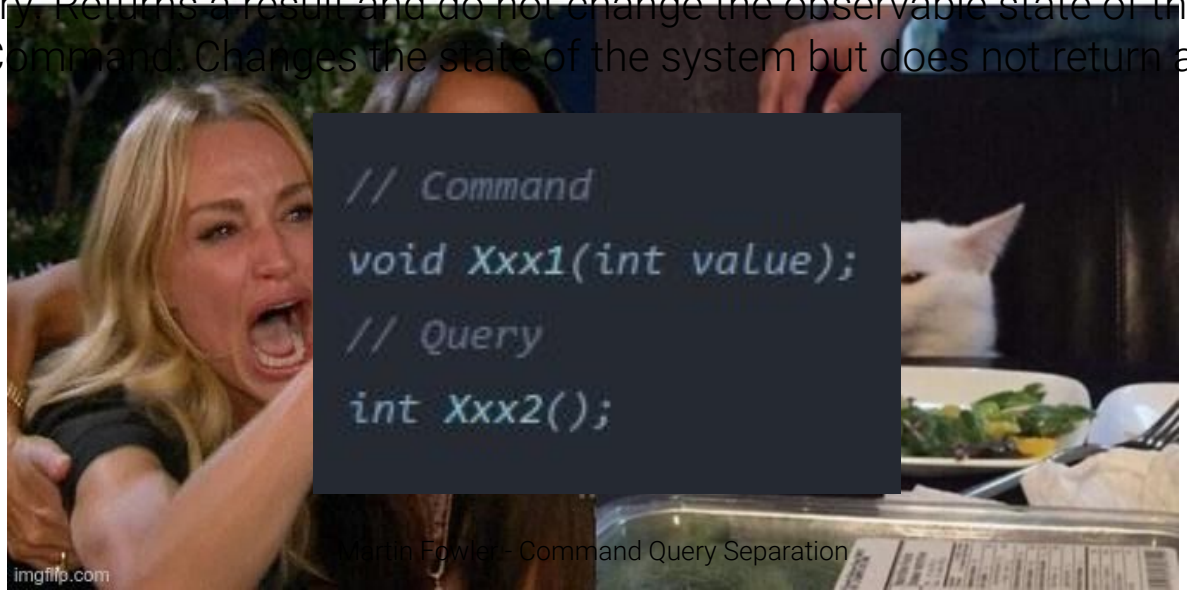
F#

```fsharp
int -> int  tr00d
let incrementValue value = value + 1
'a -> string  tr00d
let convertToMessage value = $"The value is some {value}!"
string  tr00d
let defaultMessage = "The value is none"
'a option -> string  tr00d
let matchResult content =
    match content with
    | None -> defaultMessage
    | Some value -> convertToMessage value

[<Fact>]
unit -> unit  tr00d
let ``F# equivalent!`` () =
    Some(3) : int option
    |> Option.map incrementValue : int option
    |> Option.map incrementValue : int option
    |> Option.map incrementValue : int option
    |> matchResult : string
    |> should equal "The value is some 6!"
```

# Can two paradigms coexist?

OOP **Command Query Separation** FP

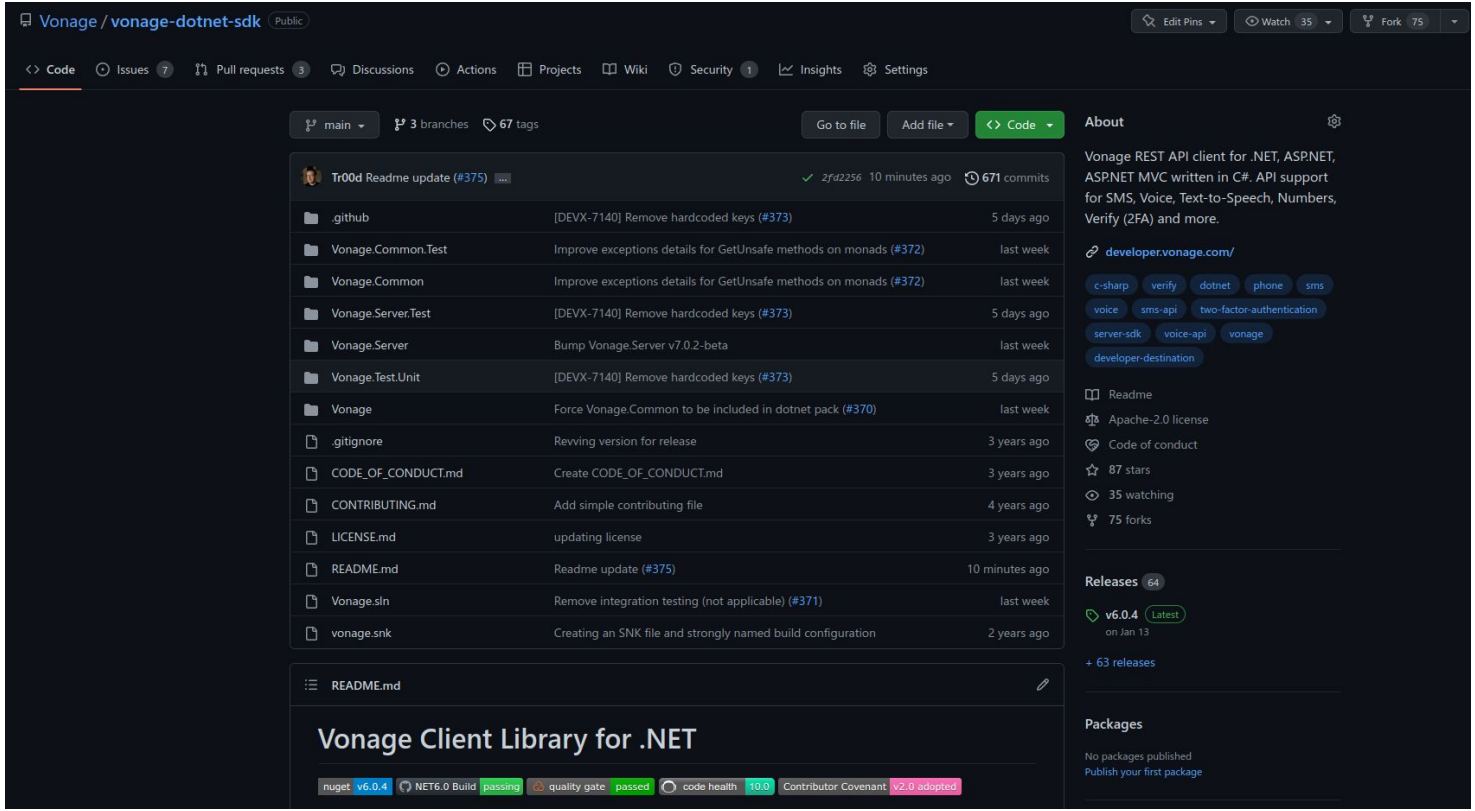
- Query: Returns a result and do not change the observable state of the system
- Command: Changes the state of the system but does not return a value




Martin Fowler - Command Query Separation

https://www.martinfowler.com/bliki/CommandQuerySeparation.html

# What do we do?

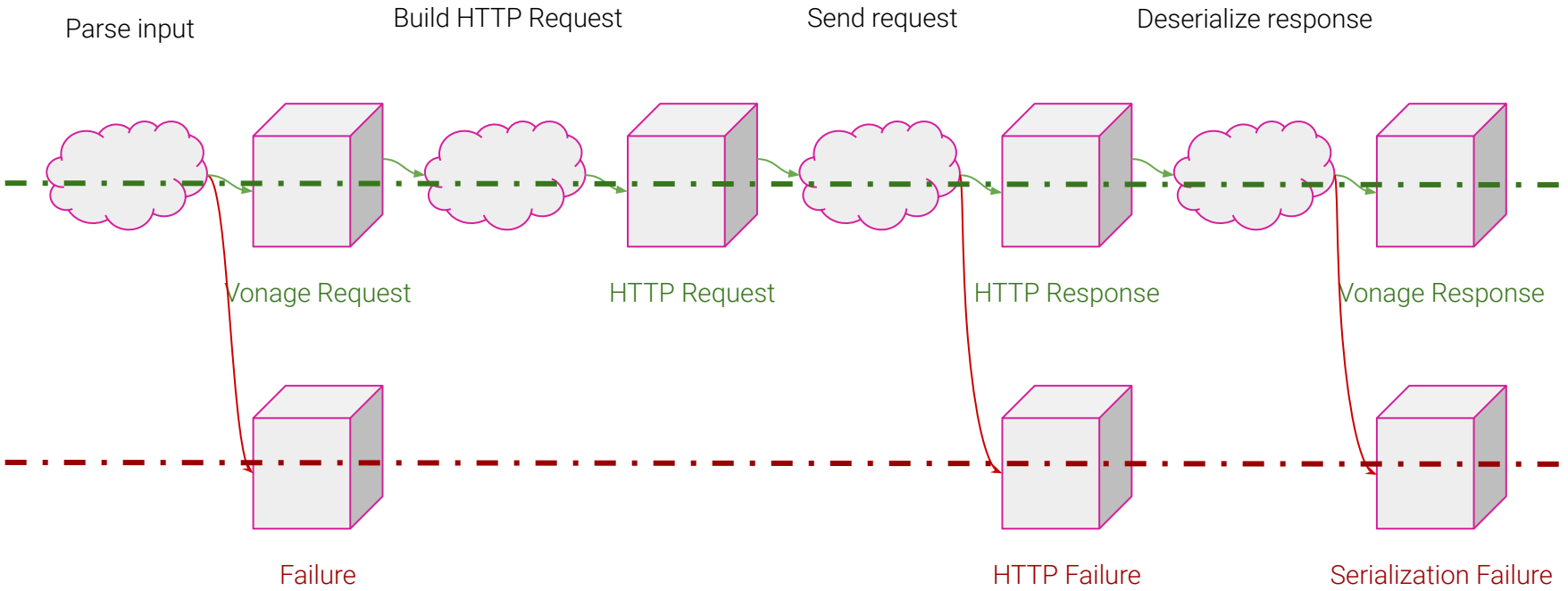

Vonage / vonage-dotnet-sdk
Public
Edit Pins
Watch 35
Fork 75
Code
Issues 7
Pull requests 3
Discussions
Actions
Projects
Wiki
Security 1
Insights
Settings
main
3 branches
67 tags
Go to file
Add file
Code
Tr00d Readme update (#375)
2fd2256 10 minutes ago
671 commits
.github [DEVX-7140] Remove hardcoded keys (#373) 5 days ago
Vonage.Common.Test Improve exceptions details for GetUnsafe methods on monads (#372) last week
Vonage.Common Improve exceptions details for GetUnsafe methods on monads (#372) last week
Vonage.Server.Test [DEVX-7140] Remove hardcoded keys (#373) 5 days ago
Vonage.Server Bump Vonage.Server v7.0.2-beta last week
Vonage.Test.Unit [DEVX-7140] Remove hardcoded keys (#373) 5 days ago
Vonage Force Vonage.Common to be included in dotnet pack (#370) last week
.gitignore Revving version for release 3 years ago
CODE_OF_CONDUCT.md Create CODE_OF_CONDUCT.md 3 years ago
CONTRIBUTING.md Add simple contributing file 4 years ago
LICENSE.md updating license 3 years ago
README.md Readme update (#375) 10 minutes ago
Vonage.sln Remove integration testing (not applicable) (#371) last week
vonage.snk Creating an SNK file and strongly named build configuration 2 years ago
README.md
Vonage Client Library for .NET
nuget v6.0.4
NET6.0 Build passing
quality gate passed
code health 10.0
Contributor Covenant v2.0 adopted
About
Vonage REST API client for .NET, ASP.NET, ASP.NET MVC written in C#. API support for SMS, Voice, Text-to-Speech, Numbers, Verify (2FA) and more.
developer.vonage.com/
c-sharp verify dotnet phone sms voice sms-api two-factor-authentication server-sdk voice-api vonage developer-destination
Readme
Apache-2.0 license
Code of conduct
87 stars
35 watching
75 forks
Releases 64
v6.0.4 Latest
on Jan 13
+ 63 releases
Packages
No packages published
Publish your first package

# Parse, don’t Validate

```
public static Result<StopArchiveRequest> Parse(Guid applicationId, Guid archiveId) =>
    Result<StopArchiveRequest>
        .FromSuccess(new StopArchiveRequest(applicationId, archiveId))
        .Bind(VerifyApplicationId)
        .Bind(VerifyArchiveId);

private static Result<StopArchiveRequest> VerifyApplicationId(StopArchiveRequest request) =>
    InputValidation.VerifyNotEmpty(request, request.ApplicationId, nameof(ApplicationId));

private static Result<StopArchiveRequest> VerifyArchiveId(StopArchiveRequest request) =>
    InputValidation.VerifyNotEmpty(request, request.ArchiveId, nameof(ArchiveId));
```

*“Parse, don’t Validate”* - Alexis King
https://lexi-lambda.github.io/blog/2019/11/05/parse-don-t-validate/

# Sending requests



# Steps

- Parse input
- Build HttpRequest
- Send request
- Deserialize response into Success/Failure depending on status code

```
await request
    .Map(this.BuildHttpRequestMessage)
    .MapAsync(value => this.client.SendAsync(value))
    .BindAsync(value => MatchResponse(value, this.ParseFailure<TResponse>, this.ParseSuccess<TResponse>));
```

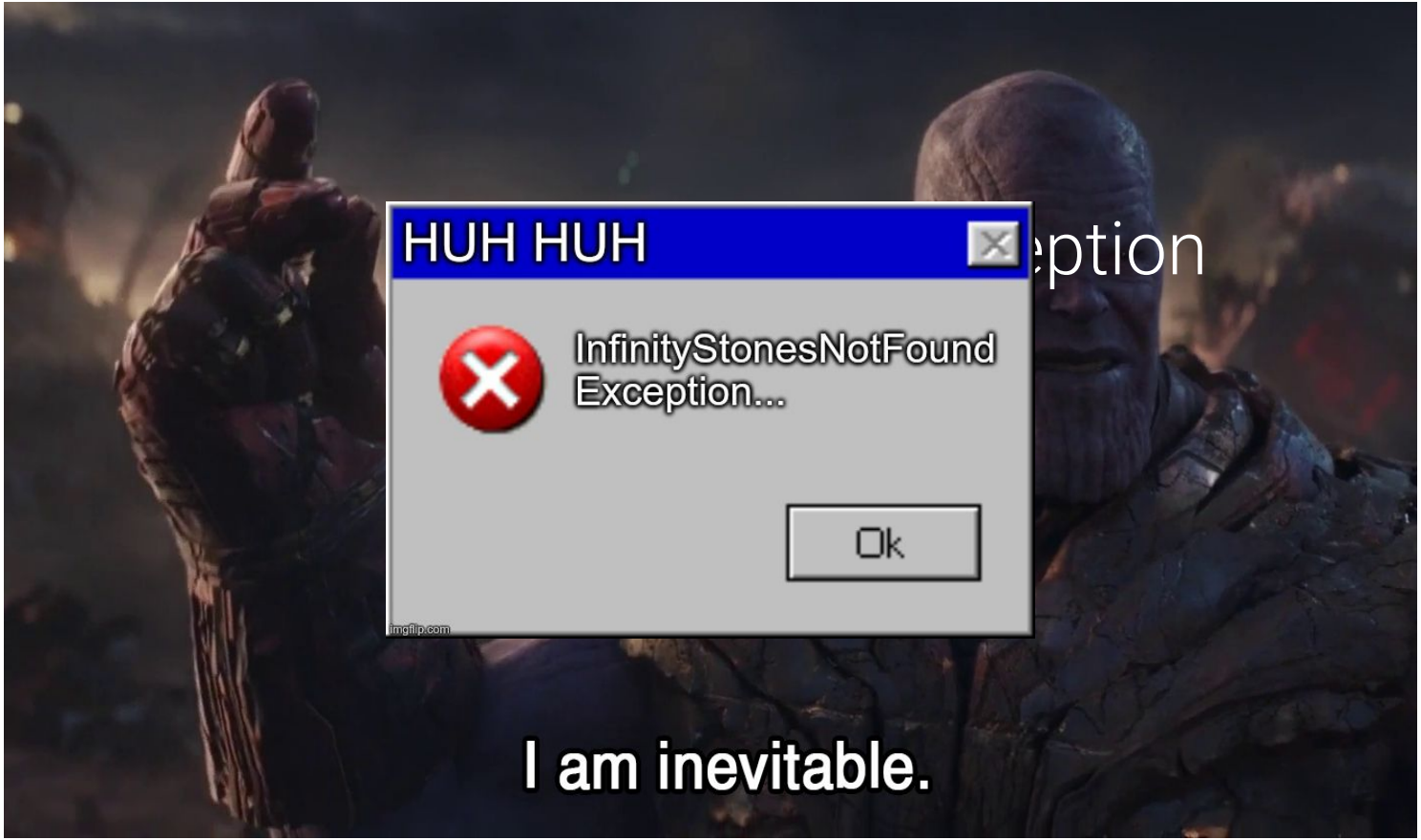
```
private static Task<Result<T>> MatchResponse<T>(
    HttpResponseMessage response,
    var result = await StopArchiveRequest
        .Parse(applicationid, archiveId)
        .BindAsync(request => client.ArchiveClient.StopArchiveAsync(request));
    result.Match(this.DoSomethingWithArchive, this.DoSomethingWithFailure);
}

private Unit DoSomethingWithArchive(Archive value) => Unit.Default;

private Unit DoSomethingWithFailure(IResultFailure failure) => Unit.Default;
        : this.CreateFailureResult<T>(response.StatusCode, responseContent);
}

private async Task<Result<T>> ParseSuccess<T>(HttpResponseMessage response)
{
    var responseContent = await response.Content.ReadAsStringAsync();
    return this.jsonSerializer
        .DeserializeObject<T>(responseContent)
        .Match(Result<T>.FromSuccess, Result<T>.FromFailure);
}
```

# What if you WANT exceptions anyway?

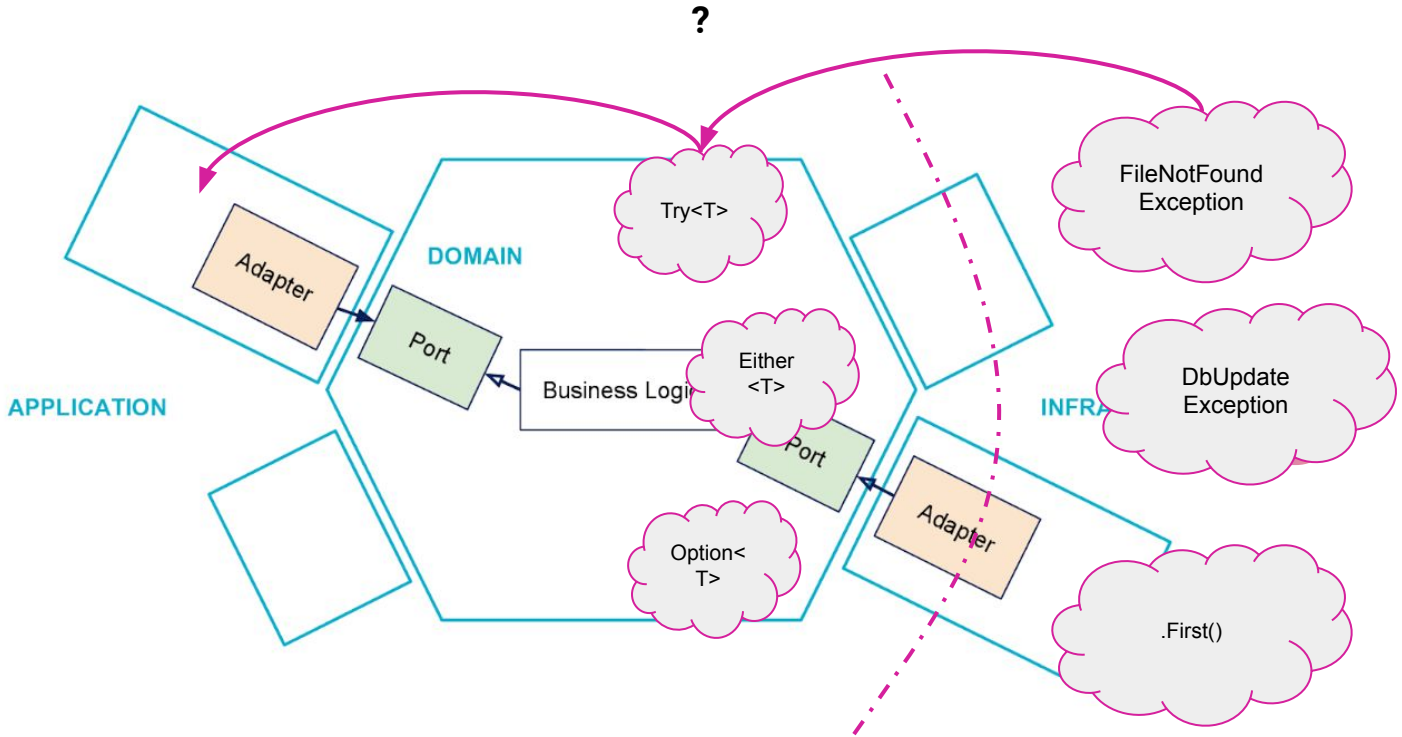
```
// Providing a function for both states
result.Match(DoSomethingWithValue, DoSomethingWithFailure);

// Receiving an exception if the monad is in the Failure state
try
{
    var value = result.GetSuccessUnsafe();
    DoSomethingWithValue(value);
}
catch (FailureStateException exception)
{
    DoSomethingWithFailure(exception.Failure);
}
```

# Getting rid of all exceptions?

Really?


HUH HUH
InfinityStonesNotFound Exception...
Ok
imgflip.com
I am inevitable.

# Real scenario with Hexagonal Architecture



# Why did I made this talk?

- Expose a viable alternative to exception handling
- Give you another tool for your developer toolbox

# Throwing exceptions or not DOES NOT MATTER

All of this for nothing?

“When I [decide to break CQS], I do so realising I’m making a trade-off, and I don’t do it lightly”

Mark Seemann - “CQS versus server-generated ids”
https://blog.ploeh.dk/2014/08/11/cqs-versus-server-generated-ids/

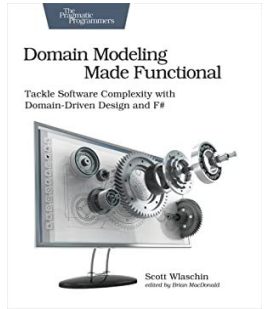
Know why you’re doing it

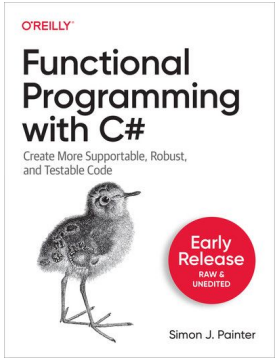
Know what compromises you make along the way

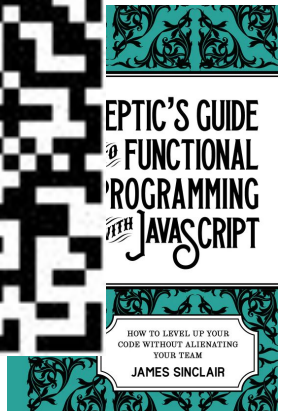
Be able to explain and defend your ideas

# Resources / Free Vonage coupon

- "*Functor, Applicatives and Monads in pictures*" - Adit Bhargava
  https://www.adit.io/posts/2013-04-17-functors,_applicatives,_and_monads_in_pictures.html#just-what-is-a-functor,-really
- "*Railway Oriented Programming Error hand*
  https://swlaschin.gitbooks.io/fsharpforfu
- "*Command Query Separation*" - Martin Fov
  https://www.martinfowler.com/bliki/Com
- Language-Ext
  https://github.com/louthy/language-ext
- "*Parse, don't Validate*" - Alexis King
  https://lexi-lambda.github.io/blog/2019/1
- Vonage .NET SDK
  https://github.com/Vonage/vonage-dotne
- "Throw exceptions… out of your codebase
  https://github.com/Tr00d/talk-throw-exce
- "CQS versus server-generated ids" - Mark
  https://blog.ploeh.dk/2014/08/11/cqs-ver
- F# for fun and profit
  https://fsharpforfunandprofit.com/
- "Saving christmas with functional C#" - Si
  https://www.thecodepainter.co.uk/blog/2
- XtremTDD - how to implement monads (f
  https://github.com/les-tontons-crafters/x
- XtremTDD - What's a monad?
  https://xtrem-tdd.netlify.app/Flavours/monads



# THANK YOU !

GAMING1

ONIRYX i'groove

PROUDLY PART OF BLACKSHEEP TRIBES

GENESIS CONSULT
IT Consultancy Services

BuSI

nsi
A CEGEKA COMPANY

MIRAHI
ミラヒ
COMPANY

delaware

[sfeir]

devday
GET INSPIRED

# Feedbacks


ANY QUESTIONS?