10 Opinions For Creating More Maintainable .NET Apps

Slides up at scottsauber.com

scottsauber

Audience Goals

- “New” perspective on a few topics
- You’re not going to agree on everything
- Take away some ideas to implement this afternoon

Who am I?

- Director of Engineering at Lean TECHniques
- Co-organizer of Iowa .NET User Group
- Blog at scottsauber.com

Agenda

- Lightning Talk approach
- Talk about my opinions for all these
  - Folder Structure
  - Don’t use IOptions Directly
  - Code Flow
  - Validation
  - ORM’s
  - Dependency Injection
  - Unit Testing and Assertions
  - DevOps
  - Feature Toggles
  - Microservices
- Pause for questions after each

Overall theme

- The point of software is to *sustainably* minimize lead time to business impact
- “Legacy software is software you have no confidence in.”

Guillermo Rauch
@rauchg

Every system tends towards complexity, slowness and difficulty
Staying simple, fast and easy-to-use is a battle that must be fought everyday

5:39 PM · Dec 26, 2016 from San Diego, CA · Twitter Web Client

642 Retweets **26** Quote Tweets **1,092** Likes

Michael Feathers
@mfeathers

Make something simple and add complexity reluctantly.

9:08 PM · Jul 6, 2017 · Twitter for iPhone

84 Retweets **1** Quote Tweet **125** Likes

Folder Structure

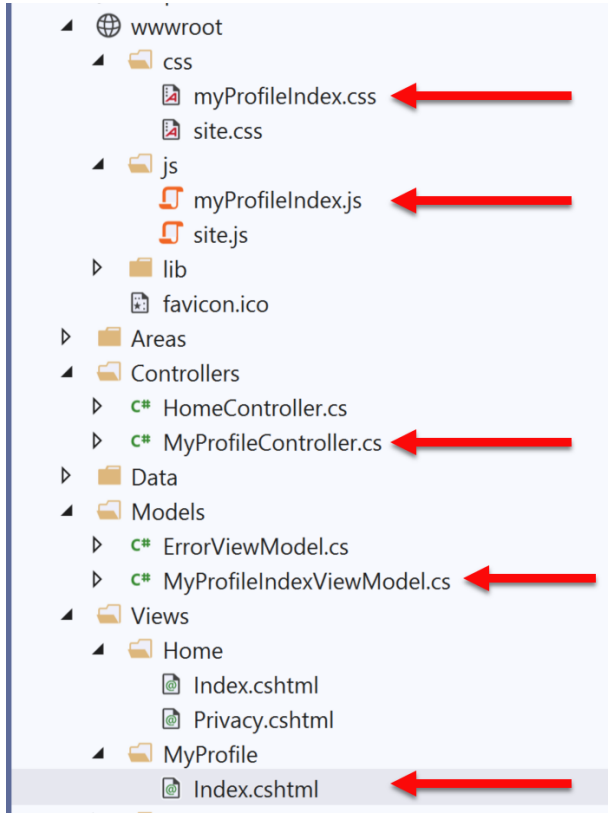
Problem: OOB MVC Folders By Responsibility

- All of these live in their own separate folders and most are required to add a new feature
  - Controllers
  - Views
  - Models
  - wwwroot/css
  - wwwroot/js
- Adds navigation friction
- Scope of a feature is scattered
- Makes it hard to add, delete or extend existing features

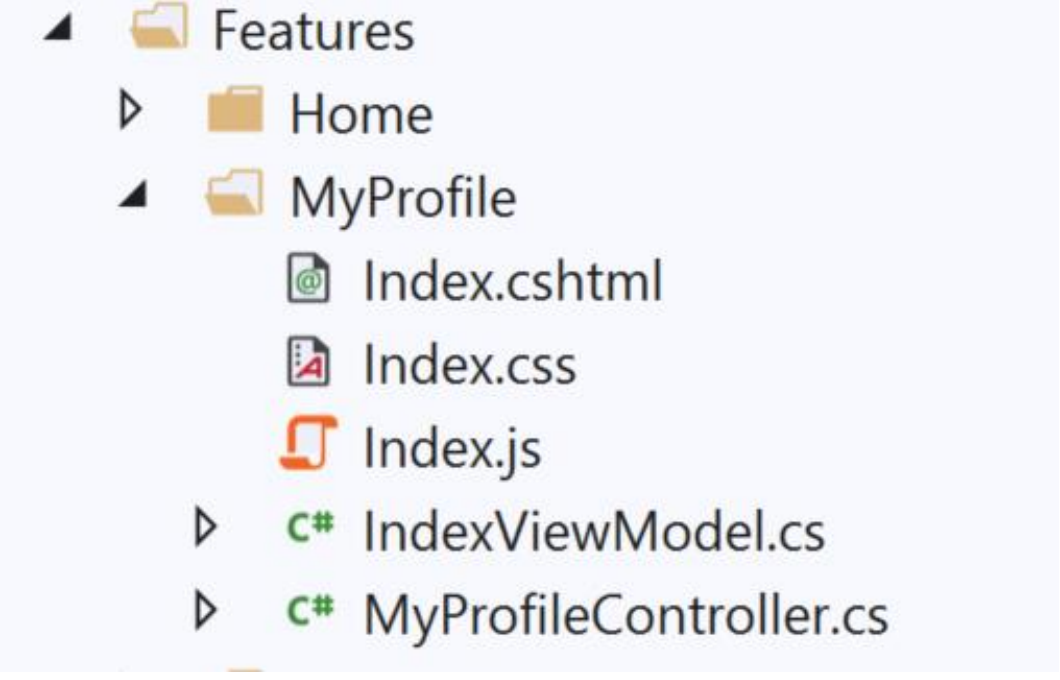
Solution: Use Feature Folders

- Grouping by Feature, not by Responsibility, results in easier maintenance
- Related things remain together (High Cohesion)

MVC out of the box:

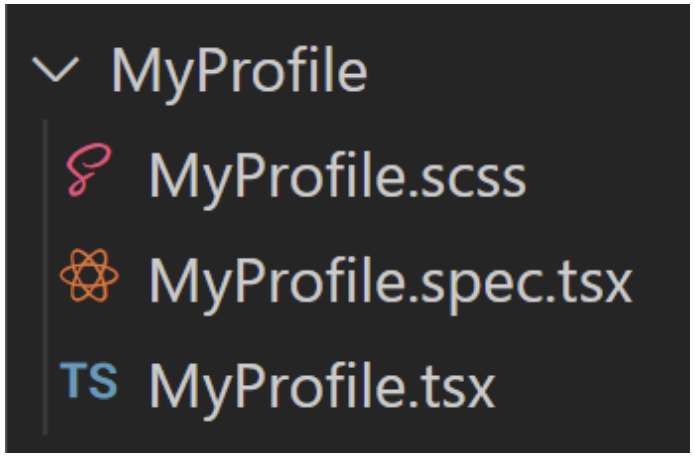


Feature Folders:

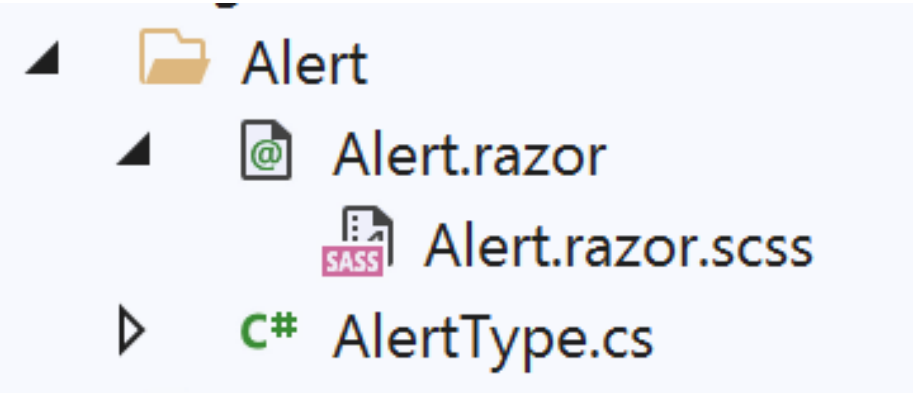


Solution: Use Feature Folders

React:



Blazor:



Feature Folder Extra Resources

- Soap analogy
- How to do this in ASP.NET Core
  - My blog post
  - Steve Smith's Blog on Feature Folders vs. Areas
- Refactoring to Vertical Slice architecture (featureFolders++)
  - Derek Comartin

Questions on Feature Folders?

Don’t use IOptions… Directly

Problem: IOptions is annoying

- Dependency on Microsoft.Extensions.Options down in other csproj's
- .Value everywhere adds friction
- Testing IOptions is slightly annoying with Options.Create()

```
public AppSettings AppSettings { get; }

public IndexModel(IOptions<AppSettings> appSettings)
{
    AppSettings = appSettings.Value;
}
```

Solution: Register your Options class directly

```
services.Configure<AppSettings>(Configuration.GetSection(key: "AppSettings"));
services.AddSingleton(registeredServices :IServiceProvider =>
    registeredServices.GetRequiredService<IOptions<AppSettings>>().Value);
```

```
public AppSettings AppSettings { get; }

public IndexModel(AppSettings appSettings)
{
    AppSettings = appSettings;
}
```

scottsauber

Questions on IOptions?

Code Smells

Structuring a method

- Happy Path always at the bottom of the method
  - Don’t want to scan for “what happens when all goes well” and find it in the middle of a method
- Use return’s instead of nested if => else

Razor Pages Template Code:

```
public async Task<IActionResult> OnPostAsync(string returnUrl = null)
{
    returnUrl = returnUrl ?? Url.Content("~/");
    if (ModelState.IsValid)
    {
        var user = new IdentityUser { UserName = Input.Email, Email = Input.Email };
        var result = await _userManager.CreateAsync(user, Input.Password);
        if (result.Succeeded)
        {
            _logger.LogInformation("User created a new account with password.");

            var code = await _userManager.GenerateEmailConfirmationTokenAsync(user);
            var callbackUrl = Url.Page("/Account/ConfirmEmail",null, new { userId = user.Id, code = code }, Request.Scheme);

            await _emailSender.SendEmailAsync(Input.Email, "Confirm your email",
                $"Please confirm your account by <a href='{HtmlEncoder.Default.Encode(callbackUrl)}'>clicking here</a>.");

            await _signInManager.SignInAsync(user, isPersistent: false);
            return LocalRedirect(returnUrl);
        }
        foreach (var error in result.Errors)
        {
            ModelState.AddModelError(string.Empty, error.Description);
        }
    }

    // If we got this far, something failed, redisplay form
    return Page();
}
```

Refactored with Happy Path At The Bottom:

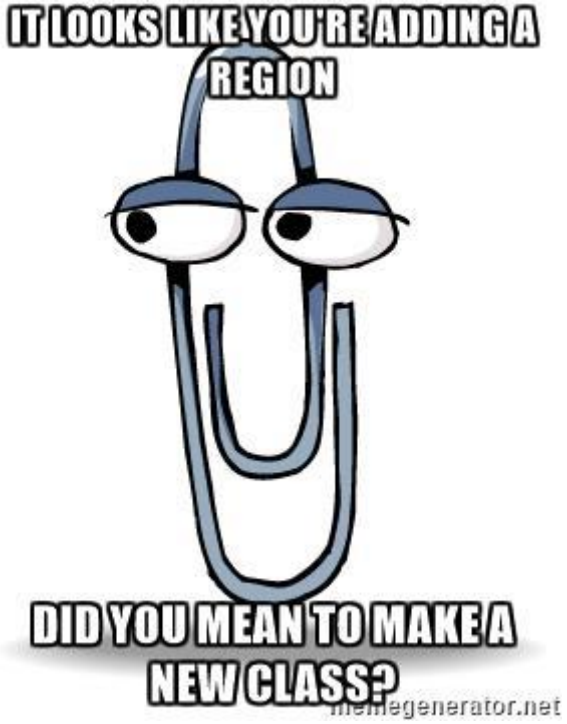
```
public async Task<IActionResult> OnPostAsync(string returnUrl = null)
{
    returnUrl = returnUrl ?? Url.Content("~/");
    if (!ModelState.IsValid)
        return Page();

    var user = new IdentityUser { UserName = Input.Email, Email = Input.Email };
    var result = await _userManager.CreateAsync(user, Input.Password);

    if (!result.Succeeded)
    {
        foreach (var error in result.Errors)
        {
            ModelState.AddModelError(string.Empty, error.Description);
        }

        return Page();
    }

    _logger.LogInformation("User created a new account with password.");

    var code = await _userManager.GenerateEmailConfirmationTokenAsync(user);
    var callbackUrl = Url.Page("/Account/ConfirmEmail", null, new { userId = user.Id, code = code }, Request.Scheme);

    await _emailSender.SendEmailAsync(Input.Email, "Confirm your email",
        $"Please confirm your account by <a href='{HtmlEncoder.Default.Encode(callbackUrl)}'>clicking here</a>.");

    await _signInManager.SignInAsync(user, isPersistent: false);
    return LocalRedirect(returnUrl);
}
```

The Indentation Proclamation

- The more indented your code is, the harder it is to follow
- Nested if’s, nested loops, etc.
- …I made this up

Code smells

- Not hard and fast rules, just my “warning light”
- Methods > 20 lines
- Classes > 200 lines
- Regions
  - You probably should’ve added a new class or method instead



Questions on Code Flow/Smells?

Validation

Validation – What's wrong with OOB Options

- Data Annotations
  - Only work well for simple scenarios
  - Hard to make custom ones
  - Hard to unit test
  - Separate annotations for each property
    - Can get "tall"
  - SRP violated
    - Model + Validation combined into one class
- Writing own Custom Validation methods
  - Lose client-side hooks Data Annotations provides if you're server rendering

Solution: Use FluentValidation

Fluent*
Validation

- Fluent interface
- Business rules are easy to maintain and read
- Easy to show a stakeholder
- Easy to test
- Integrates with ModelState.IsValid
- Same Client-Side validation as Data Annotations
- 123M downloads
- https://github.com/JeremySkinner/FluentValidation

Template Code:

```
public class RegisterViewModel
{
    [Required]
    [EmailAddress]
    [Display(Name = "Email")]
    2 references | 0 exceptions
    public string Email { get; set; }

    [Required]
    [StringLength(100, ErrorMessage = "The {0} must be at least {2} characters long.", MinimumLength = 6)]
    [DataType(DataType.Password)]
    [Display(Name = "Password")]
    1 reference | 0 exceptions
    public string Password { get; set; }

    [DataType(DataType.Password)]
    [Display(Name = "Confirm password")]
    [Compare("Password", ErrorMessage = "The password and confirmation password do not match.")]
    0 references | 0 exceptions
    public string ConfirmPassword { get; set; }
}
```

Refactored with Fluent Validation:

```
public class RegisterViewModel
{
    [Display(Name = "Email")]
    4 references | 0 exceptions
    public string Email { get; set; }

    [DataType(DataType.Password)]
    [Display(Name = "Password")]
    5 references | 0 exceptions
    public string Password { get; set; }

    [DataType(DataType.Password)]
    [Display(Name = "Confirm password")]
    1 reference | 0 exceptions
    public string ConfirmPassword { get; set; }
}
```

```
public class RegisterViewModelValidator : AbstractValidator<RegisterViewModel>
{
    0 references | 0 exceptions
    public RegisterViewModelValidator()
    {
        RuleFor(m => m.Email).NotEmpty()
            .WithMessage("Email is required.");

        RuleFor(m => m.Email).EmailAddress()
            .WithMessage("Email must be a valid email address.");

        RuleFor(m => m.Password).NotEmpty()
            .WithMessage("Password is required.");

        RuleFor(m => m.Password).MaximumLength(100)
            .WithMessage("The password cannot be longer than 100 characters.");

        RuleFor(m => m.Password).MinimumLength(6)
            .WithMessage("The password must be at least 6 characters long");

        RuleFor(m => m.ConfirmPassword).Equal(m => m.Password)
            .WithMessage("The password and confirmation password do not match.");
    }
}
```

A Rule that only exists if….

```
public class InsuranceEnrollmentValidator : AbstractValidator<InsuranceEnrollment>
{
    0 references
    public InsuranceEnrollmentValidator()
    {
        RuleFor(model => model.Age)
            .Must(age => age < 26)
            .When(model => model.IsDependent)
            .WithMessage("A dependent must be younger than 26.");
    }
}
```

Questions on Fluent Validation?

ORM’s

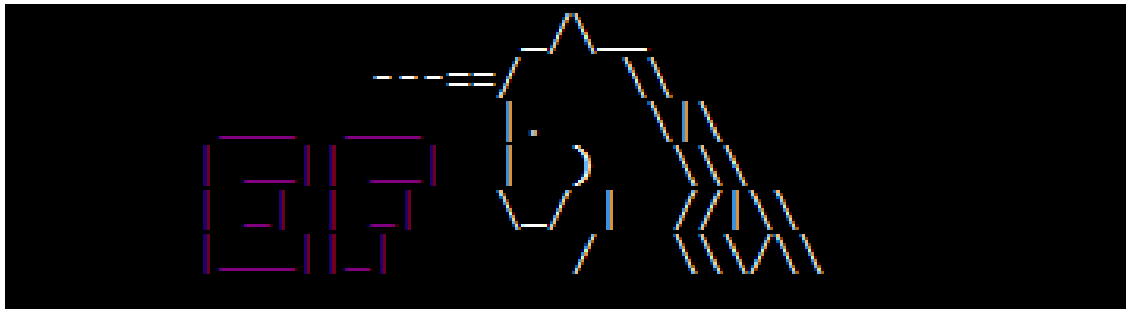
Don’t use raw ADO - Use an ORM

Entity Framework – Full ORM

- Pros
  - Developer Productivity
  - Compile-time safety with LINQ Queries
  - Extremely quick to add new CRUD operations
  - Built in Unit of Work
  - Migration support
- Cons
  - Less performant
  - Less control over the queries generated
  - Heavier

Dapper – Micro ORM

- Pros
  - Performance near ADO
  - More control over the queries
  - Extremely simple to setup
  - Stack Overflow beta tests
- Cons
  - SQL strings
  - Less features than EF



ORM usage comparison

Entity Framework:

```
public IEnumerable<Customer> GetCustomersByState(string state)
{
    var dbContext = new ApplicationDbContext();
    return dbContext.Customers.Where(c:Customer => c.State == state).ToList();
}
```

Dapper:

```
public IEnumerable<Customer> GetCustomersByState(string state)
{
    var sqlConnection = new SqlConnection();

    var sql = "SELECT * FROM Customers WHERE State=@State";

    var parameters:{State} = new { State = state };

    return sqlConnection.Query<Customer>(sql, parameters);
}
```

Other ORM things

- Both manage connection lifetimes
- Both give you SQL Injection protection
- I use both depending on the job
- DON’T WRITE YOUR OWN ORM

Questions on ORM’s?

Dependency Injection

Dependency Injection and IoC containers

- DI/IoC helps you loosely couple your apps
- Easily swap out different implementations
  - Switching directions
  - Running locally vs in the cloud
- Lets you unit test anything if done correctly
- I use Microsoft.Extensions.DependencyInjection
  - Scrutor if you want auto-registration

Common DI Pitfalls

- “New Is Glue”
- “Static Cling”
- DateTime.UtcNow (and variants)
  - Instead inject in IClock or have method take in DateTime? and default to Now

Questions on Dependency Injection?

Automated Tests

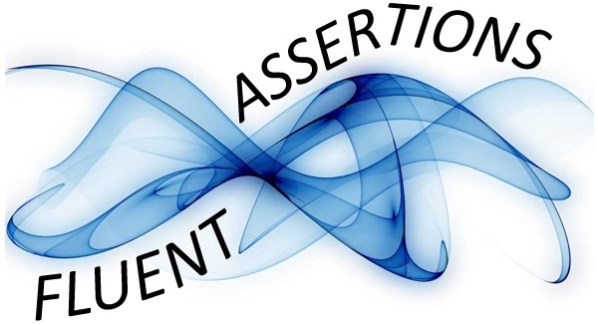
Automated Testing with xUnit

- You should be writing automated tests
  - Exposes holes in your architecture
  - Proven to be faster long-term
    - Make changes quickly and confidently because have a regression test suite
- Use xUnit or NUnit
  - Just not MSTest which is wayyy more verbose and has less features
- xUnit used by ASP.NET team
- I used to use NUnit and switched to xUnit
- NUnit more boilerplate
  - No [TestFixture] in xUnit
  - No [SetUp] method just use a constructor in xUnit

Problem: OOB Assertion Libraries Annoy Me

- Assert.Equal(value, value)
- Hard to remember that it’s Assert.Equal(expected, actual)
  - Yes I know analyzers are baked into xUnit to help here
- Can lead to funky looking assertion failures if you flip them

Solution: Use FluentAssertions for assertions



- Adds a Should() extension method to object
  - result.Should().Be(0);
  - Easier to read than Assert.Equal(0, result);
- Should().BeEquivalentTo();
  - actualCustomer.Should().BeEquivalentTo(expectedCustomer);
- 100M+ downloads

Chekhov’s Gun

scottsauber

Chekhov’s Gun

“Remove everything not relevant to the story. If a rifle is hanging on the wall in chapter 1, it must go off in a later chapter. Otherwise it shouldn’t be there.”

- Anton Chekhov

scottsauber

Chekhov’s Gun Applied to Arrange

```
[Fact]
public void ValidateShouldReturnErrorWhenLastNameIsEmpty()
{
    var customer = new Customer
    {
        FirstName = "SpongeBob",
        LastName = "",
        Address = "123 Pineapple",
        BirthDate = new DateOnly( year: 1999,  month: 5,  day: 1),
    };

    var result = new CustomerValidator().Validate(customer);

    result.Errors.Should().Contain(error :ValidationFailure => error.ErrorMessage == "Last Name is required.");
}
```

scottsauber

Chekhov’s Gun Applied to Arrange

```
[Fact]
public void ValidateShouldReturnErrorWhenLastNameIsEmpty()
{
    var customer = CreateValidCustomer();
    customer.LastName = "";

    var result = new CustomerValidator().Validate(customer);

    result.Errors.Should().Contain(error :ValidationFailure => error.ErrorMessage == "Last Name is required.");
}
```

scottsauber

Chekhov’s Gun Applied to Arrange

```
[Fact]
public void ValidateShouldReturnErrorWhenLastNameIsEmpty()
{
    _customer.LastName = "";

    var result = new CustomerValidator().Validate(_customer);

    result.Errors.Should().Contain(error:ValidationFailure => error.ErrorMessage == "Last Name is required.");
}
```

Chekhov's Gun Applied to Arrange

```
[Fact]
public void ValidateShouldReturnErrorWhenLastNameIsEmpty()
{
    _customer.LastName = "";

    var result = _customerValidator.Validate(_customer);

    result.Errors.Should().Contain(error:ValidationFailure => error.ErrorMessage == "Last Name is required.");
}
```

Comparison

```
[Fact]
public void ValidateShouldReturnErrorWhenLastNameIsEmpty()
{
    var customer = new Customer
    {
        FirstName = "SpongeBob",
        LastName = "",
        Address = "123 Pineapple",
        BirthDate = new DateOnly(year: 1999, month: 5, day: 1),
    };

    var result = new CustomerValidator().Validate(customer);

    result.Errors.Should().Contain(error :ValidationFailure => error.ErrorMessage == "Last Name is required.");
}
```

```
[Fact]
public void ValidateShouldReturnErrorWhenLastNameIsEmpty()
{
    _customer.LastName = "";

    var result = _customerValidator.Validate(_customer);

    result.Errors.Should().Contain(error :ValidationFailure => error.ErrorMessage == "Last Name is required.");
}
```

scottsauber

Questions on Automated Testing?

DevOps

The Expert Beginner
@ExpertBeginner1

Follow

The most important thing you can do is have well-defined handoff procedures between Dev and DevOps and between DevOps and Ops.

9:24 AM - 14 Sep 2017

43 Retweets **54** Likes

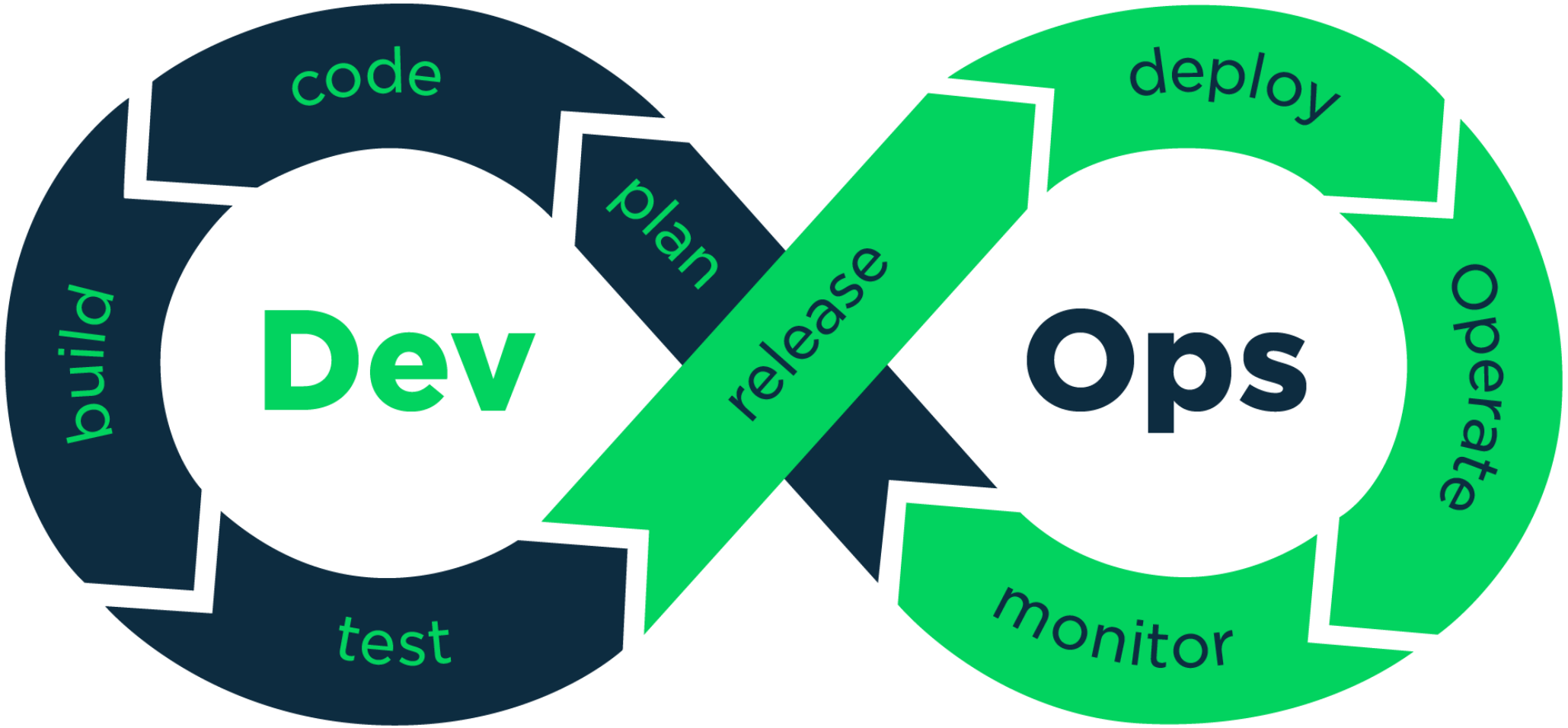
5 43 54

What is DevOps

“DevOps is the union of people, process, and products to enable continuous delivery of value to our end users.”

- Donovan Brown

DevOps Pipeline



Continuous Integration and Delivery

- Continuous Integration is a practice not a server
- Build Server
  - Automated builds running on an independent build server
  - Automatically running automated tests
  - Creates an artifact
- Continuous Delivery
  - Takes the artifact from the build server and deploys it to each enviornment
  - Automated configuration of infrastructure (Azure, AWS, GCP, on-prem, etc)
- These two together will be transformational for you and your company

CI + CD enables

- Consistency
  - Machine repeats the same steps over and over again
- Consistency promotes Confidence
  - Passes build
  - Passes automated tests
  - Let’s deploy it to Prod!
- Confidence enables agility
  - Mid-day deployments are no big deal

Continuous Deployment

- Deploying to every environment on every push to main
- What’s stopping you from deploying on green?
- Usually more automated tests
- Add them until confidence exists

Questions on DevOps?

Feature Toggles

Feature Toggles

- No long lived branches
- No environment branches
- Trunk Based Development
  - Commit to main instead!
- Behind an if statement
- Reduce merge conflicts
- Simplest way: config files – Microsoft.FeatureManagement
  - appsettings.{Environment}.json
- Commit to cleaning up after it’s live
- De-couples deployments from releases

scottsauber

No environment branches

Gitflow Workflow

Gitflow is a legacy Git workflow that was originally a disruptive and novel strategy for managing Git branches. Gitflow has fallen in popularity in favor of trunk-based workflows, which are now considered best practices for modern continuous software development and DevOps practices. Gitflow also can be challenging to use with CI/CD. This post details Gitflow for historical purposes.

https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow

Questions on Feature Toggles?

Microservices

What do I mean by microservices?

- When I say Microservices I mean some out of process call of some sort (web service, queue, etc.)

Microservices Add Operational Complexity

- Health Checks
  - Is it running? Are the dependencies ok?
- Service Discovery
- Perf is worse
- Traceability with Correlation ID's

Microservices Add Development Complexity

- HTTP call (most likely) instead of a method call
- If calling from C#, likely want a client library wrapper around HttpClient sending JSON
- Dev environment needs multiple projects running
- Cross-cutting things (users, logging, common package versions, etc.)
- Network boundaries don’t magically make your code better
- Did you draw the right domain boundaries?
- Be wary of two-phased commits

It’s not all bad…

- Benefits
  - Iterate on microservices separate from other app (SRP)
  - Reuse
  - Scale independently
  - Easy to reason about in small chunks
- You are not Google/Facebook/Netflix/Amazon
- Use where it makes sense:
  - Needs scale
  - High CPU usage
  - Isolate an annoying dependency
  - Separate teams
  - Hard costs is a priority over engineering costs

My favorite definition of Microservices

“Loosely coupled service oriented architecture with bounded contexts.”

- Adrian Cockroft

Boundaries Boundaries Boundaries

- “Almost all the successful microservice stories have started with a monolith that got too big and was broken up
  Almost all the cases where I've heard of a system that was built as a microservice system from scratch, it has ended up in serious trouble.” – Martin Fowler

- “I remain convinced that it is much easier to partition an existing, "brownfield" system than to do so up front with a new, greenfield system.” – Sam Newman

Questions on Microservices?

BONUS

Bonus 11th Opinion

- Build/Compiler warnings don't exist in my world
  - Error or Nothing

```
<PropertyGroup>
    <TargetFramework>net6.0</TargetFramework>
    <Nullable>enable</Nullable>
    <TreatWarningsAsErrors>true</TreatWarningsAsErrors>
</PropertyGroup>
```

Real benefits of these practices

- 1 web app went from deploying to Prod 20x a year to deploying to Prod over 500x a year
- Faster delivery of value to users

Closing

- Hope you got at least one idea out of this.
- You likely don’t agree with everything I just said.
- Focus on what matters
  - xUnit vs NUnit – who cares?
  - Why > How

Questions?

Slides up at scottsauber.com

scottsauber

Thanks!

Slides up at scottsauber.com

scottsauber