

2ГИС

Как мы перешли на
`Microsoft.Extensions.Configuration`
и стало хорошо

Про меня

Я Андрей

Работаю в 2ГИС

Пишу на C#

Выступаю первый раз



Про команду

Мы занимаемся разработкой внутреннего продукта

Используем C# разных версий

Что-то большое, старое и на .NET Framework'e

Что-то поменьше, новее и уже на .NET

Как мы конфигурируем .NET Framework приложения

Как мы конфигурируем .NET Framework приложения



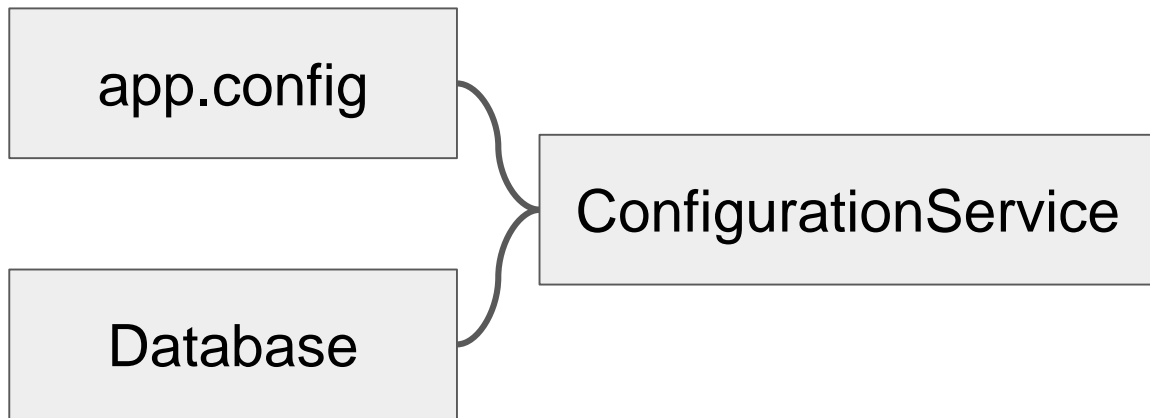
app.config

Как мы конфигурируем .NET Framework приложения

app.config

Database

Как мы конфигурируем .NET Framework приложения



Как мы конфигурируем .NET Framework приложения



Как мы конфигурируем .NET Framework приложения

Как мы конфигурируем .NET Framework приложения

app.config

- путь до базы
- адреса сервисов
- настройки логирования

Как мы конфигурируем .NET Framework приложения

app.config

- путь до базы
- адреса сервисов
- настройки логирования

База данных

- Cron джобы
- Переменные
- Фича флаги

Как мы конфигурировали .NET приложения

Как мы конфигурировали .NET приложения

appsettings.json

Как мы конфигурировали .NET приложения

appsettings.json

ENV

Как мы конфигурировали .NET приложения

Как мы конфигурировали .NET приложения

```
public class ServiceConfigs
{
    public string Host { get; }
    public string User { get; }
    public string Token { get; }

    public ServiceConfigs()
    {
        Host = Environment.GetEnvironmentVariable("@HOST");
        User = Environment.GetEnvironmentVariable("USER");
        Token = Environment.GetEnvironmentVariable("TOKEN");
    }
}
```

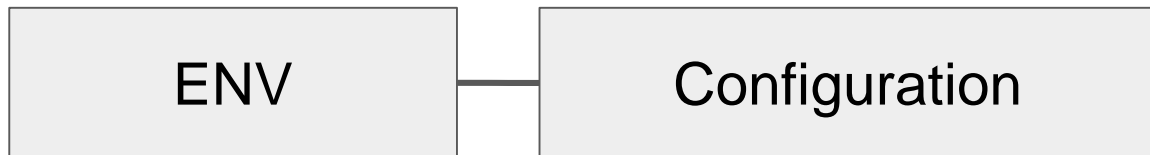

Как мы конфигурировали .NET приложения

Как мы конфигурировали .NET приложения

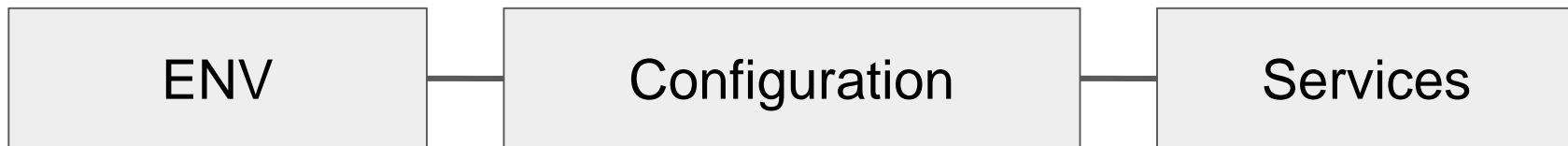


ENV

Как мы конфигурировали .NET приложения



Как мы конфигурировали .NET приложения



Назревают проблемы

Назревают проблемы

```
[Test]
public void DoWork_ShouldReturnTrue_ThenWorkIsValid()
{
    var config = new ServiceConfigs();

    var service = new Service(config);
    // Act
    // Assert
}
```

Назревают проблемы

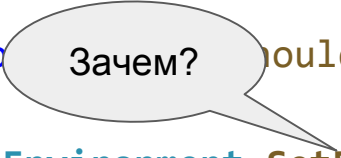
```
[Test]
public void DoWork_ShouldReturnTrue_ThenWorkIsValid()
{
    Environment.SetEnvironmentVariable("PERIOD", "10");
    var config = new ServiceConfigs();

    var service = new Service(config);
    // Act
    // Assert
}
```

Назревают проблемы

```
[Test]
public void ShouldReturnTrue_ThenWorkIsValid()
{
    Environment.SetEnvironmentVariable("PERIOD", "10");
    var config = new ServiceConfigs();

    var service = new Service(config);
    // Act
    // Assert
}
```



Назревают проблемы

```
[Test]
public void ShouldReturnTrue_ThenWorkIsValid()
{
    Environment.SetEnvironmentVariable("PERIOD", "10");
    var config = new ServiceConfigs();

    var service = new Service(config);
    // Act
    // Assert
}
```

Зачем?

Какая связь?

Назревают проблемы

```
[Test]
public void ShouldReturnTrue_ThenWorkIsValid()
{
    Environment.SetEnvironmentVariable("PERIOD", "10");
    var config = new ServiceConfigs();

    var service = new Service(config);
    // Act
    // Assert
}
```

Зачем?

Какая связь?

Почему
именно
PERIOD?

Назревают проблемы

```
[Test]
public void ShouldReturnTrue_ThenWorkIsValid()
{
    Environment.SetEnvironmentVariable("PERIOD", "10");
    var config = new ServiceConfigs();
    Service = new Service(config);
    // Assert
}
```

Зачем?

Какая связь?

Порядок
важен?

Почему
именно
PERIOD?

Обсуждаем проблемы



Выделяем проблемы

- Ручной маппинг

Выделяем проблемы

```
if (int.TryParse(Environment.GetEnvironmentVariable(@"MAX_AGE_DAYS"), out var maxDays))
{
    MaxAgeDays = maxDays;
}

if (int.TryParse(Environment.GetEnvironmentVariable(@"DELAY_MINUTES"), out var delayValue))
{
    DelayMinutes = delayValue;
}

// и еще очень много подобного кода
```

Выделяем проблемы

- Ручной маппинг
- Никакой структуры

Выделяем проблемы

```
DatabaseOne = Environment.GetEnvironmentVariable("DATABASE_ONE");
DatabaseTwo = Environment.GetEnvironmentVariable("DATABASE_TWO");
DatabaseFour = Environment.GetEnvironmentVariable("DATABASE_FOUR");
OneApiUrl = Environment.GetEnvironmentVariable(@"ONE_API_URL");
TwoApiUrl = Environment.GetEnvironmentVariable(@"TWO_API_URL");
AdfsHost = Environment.GetEnvironmentVariable(@"ADFS_HOST");
AdfsTenantId = Environment.GetEnvironmentVariable(@"ADFS_TENANT_ID");
AdfsResource = Environment.GetEnvironmentVariable(@"ADFS_RESOURCE");
AdfsAppId = Environment.GetEnvironmentVariable(@"ADFS_APP_ID");
AdfsAppKey = Environment.GetEnvironmentVariable(@"ADFS_APP_KEY");
ServiceApiUrl = Environment.GetEnvironmentVariable(@"SERVICE_API_URL");
// и еще
// и еще
// и еще...
```


Выделяем проблемы

- Ручной маппинг
- Никакой структуры
- Отсутствие валидации

Выделяем проблемы

- Ручной маппинг
- Никакой структуры
- Отсутствие валидации
- Коммиты `launchSettings`

Выделяем проблемы

- Ручной маппинг
- Никакой структуры
- Отсутствие валидации
- Коммиты `launchSettings`
- Ограниченный набор источников значений

Ищем решения

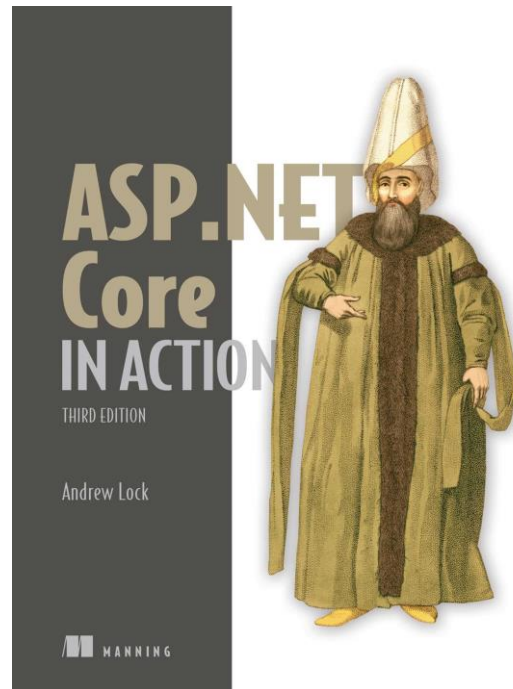
Ищем решения

- Написать все самим

Ищем решения

- Написать все самим
- Использовать подход предлагаемый Microsoft'ом

Ищем решения



Ищем решения

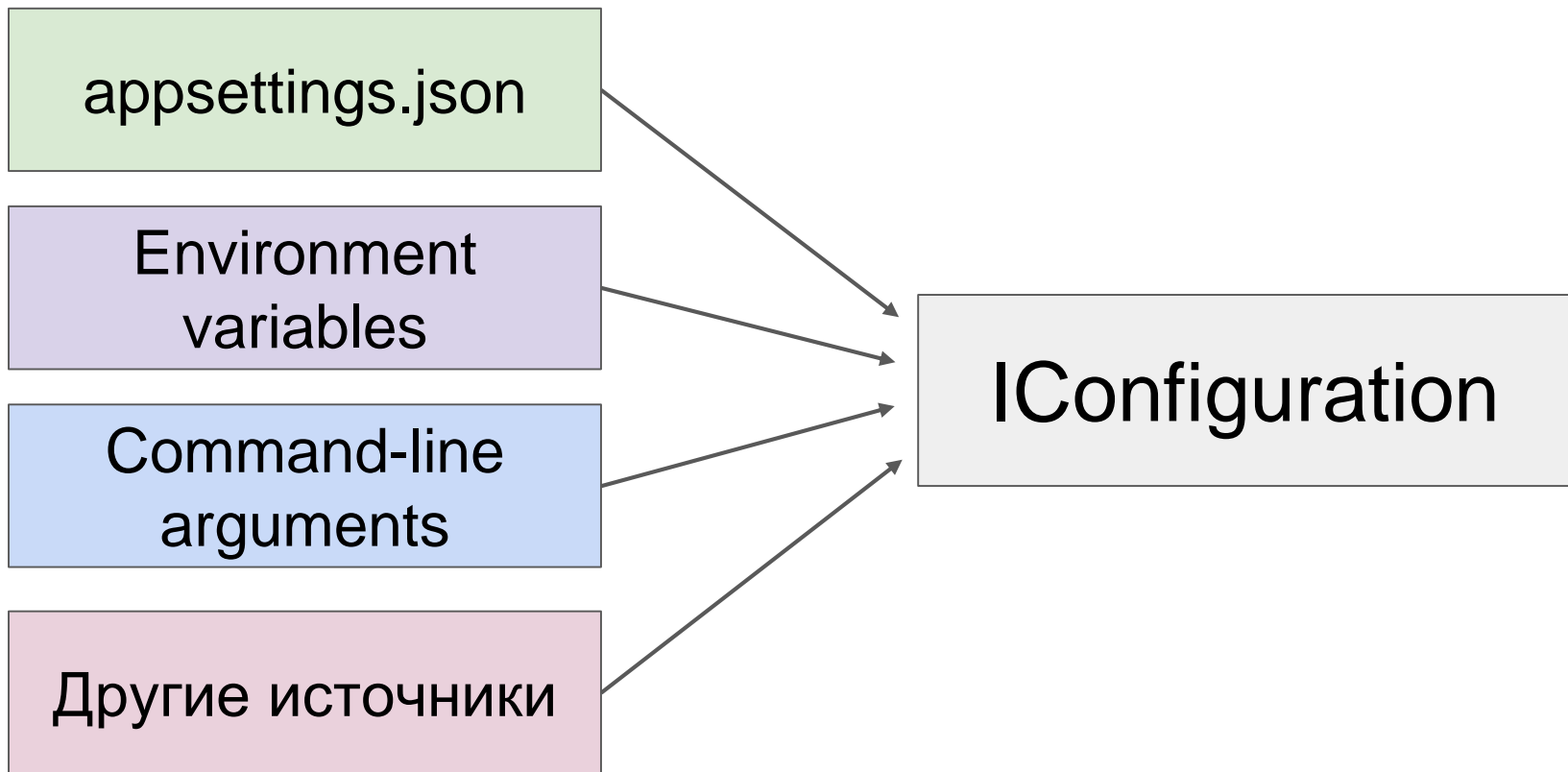
- Написать все самим
- Использовать подход предлагаемый Microsoft'ом
- Использовать сторонние библиотеки

Ищем решения

- Написать все самим
- Использовать подход предлагаемый Microsoft'ом
- Использовать сторонние библиотеки

Применение решения

Применение решения



Источники конфигураций

Источники конфигураций

- MemoryConfigurationProvider

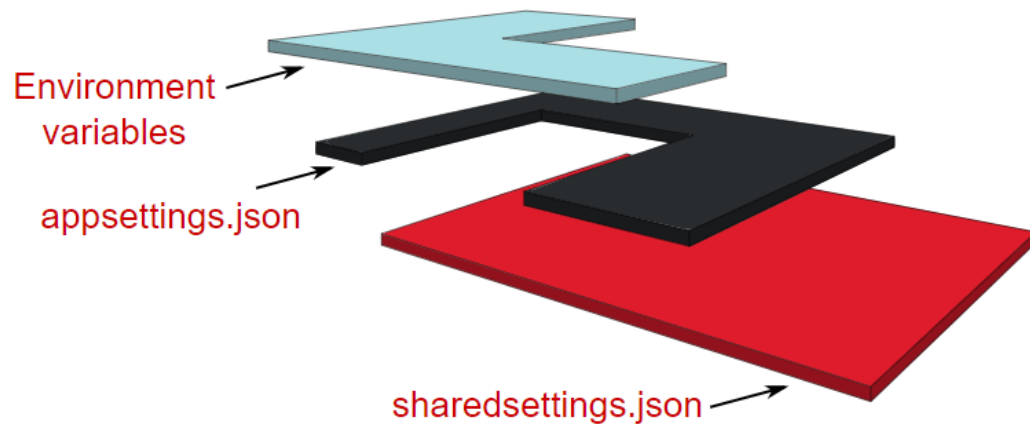
Источники конфигураций

- `MemoryConfigurationProvider`
- `EnvironmentVariablesConfigurationProvider`

Источники конфигураций

- `MemoryConfigurationProvider`
- `EnvironmentVariablesConfigurationProvider`
- `JsonConfigurationProvider`

Источники конфигураций



Источники конфигураций

Источники конфигураций

appsettings.json

```
{  
  "VeryImportantSetting": "This is a very important setting!"  
}
```

Источники конфигураций

appsettings.json

```
{  
  "VeryImportantSetting": "This is a very important setting!"  
}
```

ENV

```
"environmentVariables": {  
  "MY_VeryImportantSetting": "appsettings lie!"  
}
```

Источники конфигураций

```
builder.Configuration.AddJsonFile("appsettings.json");
```

Источники конфигураций

```
builder.Configuration.AddJsonFile("appsettings.json");  
builder.Configuration.AddEnvironmentVariables("MY_");
```

Источники конфигураций

```
builder.Configuration.AddJsonFile("appsettings.json");
```

```
builder.Configuration.AddEnvironmentVariables("MY_");
```

```
app.MapGet("/config", (IConfiguration configuration) =>  
    {  
        var config = (configuration as IConfigurationRoot).GetDebugView();  
        return config;  
    });
```

Источники конфигураций

```
builder.Configuration.AddJsonFile("appsettings.json");
```

```
builder.Configuration.AddEnvironmentVariables("MY_");
```

```
app.MapGet("/config", (IConfiguration configuration) =>
{
    var config = (configuration as IConfigurationRoot).GetDebugView();
    return config;
});
```

Источники конфигураций

```
builder.Configuration.AddJsonFile("appsettings.json");
```

```
builder.Configuration.AddEnvironmentVariables("MY_");
```

```
app.MapGet("/config", (IConfiguration configuration) =>
{
    var config = (configuration as IConfigurationRoot).GetDebugView();
    return config;
});
```

VeryImportantSetting=appsettings lie!

(EnvironmentVariablesConfigurationProvider Prefix: 'MY_')

Источники конфигураций

```
builder.Configuration.AddJsonFile("appsettings.json");
```

```
builder.Configuration.AddEnvironmentVariables("MY_");
```

```
app.MapGet("/config", (IConfiguration configuration) =>  
{  
    var config = (configuration as IConfigurationRoot).GetDebugView();  
    return config;
```



Ключ

VeryImportantSetting=appsettings lie!

(EnvironmentVariablesConfigurationProvider Prefix: 'MY_')

Источники конфигураций

```
builder.Configuration.AddJsonFile("appsettings.json");
```

```
builder.Configuration.AddEnvironmentVariables("MY_");
```

```
app.MapGet("/config", (IConfiguration configuration) =>  
{  
    var config = (configuration as IConfigurationRoot).GetDebugView();  
    return config;  
})
```

Ключ

Значение

VeryImportantSetting=appsettings lie!

(EnvironmentVariablesConfigurationProvider Prefix: 'MY_')

Источники конфигураций

```
builder.Configuration.AddJsonFile("appsettings.json");
```

```
builder.Configuration.AddEnvironmentVariables("MY_");
```

```
app.MapGet("/config", (IConfiguration configuration) =>
{
    var config = (configuration as IConfigurationRoot).GetDebugView();
    return config;
})
```

Ключ

Значение

Поставщик

VeryImportantSetting=appsettings lie!

(EnvironmentVariablesConfigurationProvider Prefix: 'MY_')

Источники конфигураций

```
builder.Configuration.AddEnvironmentVariables("MY_");
```

Источники конфигураций

```
builder.Configuration.AddEnvironmentVariables("MY_");  
builder.Configuration.AddJsonFile("appsettings.json");
```

Источники конфигураций

```
builder.Configuration.AddEnvironmentVariables("MY_");  
builder.Configuration.AddJsonFile("appsettings.json");  
  
builder.Configuration.GetDebugView();
```

Источники конфигураций

```
builder.Configuration.AddEnvironmentVariables("MY_");  
builder.Configuration.AddJsonFile("appsettings.json");
```

```
builder.Configuration.GetDebugView();
```

```
VeryImportantSetting=This is a very important setting!  
(JsonConfigurationProvider for 'appsettings.json' (Required))
```

Источники конфигураций

```
builder.Configuration.AddEnvironmentVariables("MY_");  
builder.Configuration.AddJsonFile("appsettings.json");
```

```
builder.Configuration.GetDebugView();
```

```
VeryImportantSetting=This is a very important setting!  
(JsonConfigurationProvider for 'appsettings.json' (Required))
```


Источники конфигураций

```
builder.Configuration.AddEnvironmentVariables("MY_");  
builder.Configuration.AddJsonFile("appsettings.json");
```

```
builder.Configuration.GetDebugView();
```

```
VeryImportantSetting=This is a very important setting!  
(JsonConfigurationProvider for 'appsettings.json' (Required))
```

Источники конфигураций

```
builder.Configuration.AddEnvironmentVariables("MY_");  
builder.Configuration.AddJsonFile("appsettings.json");
```

```
builder.Configuration.GetDebugView();
```

```
VeryImportantSetting=This is a very important setting!  
(JsonConfigurationProvider for 'appsettings.json' (Required))
```

Secret Manager

Secret Manager

```
dotnet user-secrets init
```

Secret Manager

```
dotnet user-secrets init
```



```
<PropertyGroup>
```

```
  <UserSecretsId>1de4633d-d945-4b73-8f0f-d72973fb04f4</UserSecretsId>
```

```
</PropertyGroup>
```



Secret Manager

```
dotnet user-secrets init
```

```
■
```

```
<PropertyGroup>
```

```
  <UserSecretsId>MyServiceSecretFolder</UserSecretsId>
```

```
</PropertyGroup>■
```

Secret Manager

```
dotnet user-secrets set "SuperSecretValue" "12345"
```

Secret Manager

```
dotnet user-secrets set "Article:Title" "Awesome story"  
dotnet user-secrets set "Article:Author" "Joe"
```


Secret Manager

```
{  
  "Article": {  
    "Title": "Awesome story",  
    "Author": "Joe"  
  }  
}
```

Источники конфигураций

Источники конфигураций

```
internal sealed class CustomConfigProvider : ConfigurationProvider
{
    public override void Load()
    {
        // получите данные
        // сложите их в поле Data
        // вы прекрасны!
    }
}
```

Источники конфигураций

```
public class DatabaseConfigurationProvider : ConfigurationProvider
{
    private readonly IRepositoryFactory _repositoryFactory;

    public DatabaseConfigurationProvider(IRepositoryFactory repositoryFactory)
    {
        _repositoryFactory = repositoryFactory;
    }

    public override void Load()
    {
        using var repository = _repositoryFactory.Create();
        Data = repository
            .GetTable<JobConfiguration>()
            .ToDictionary(v => v.Key, k => k.Value);
    }
}
```

Источники конфигураций

```
public class DatabaseConfigurationProvider : ConfigurationProvider
{
    private readonly IRepositoryFactory _repositoryFactory;

    public DatabaseConfigurationProvider(IRepositoryFactory repositoryFactory)
    {
        _repositoryFactory = repositoryFactory;
    }

    public override void Load()
    {
        using var repository = _repositoryFactory.Create();
        Data = repository
            .GetTable<JobConfiguration>()
            .ToDictionary(v => v.Key, k => k.Value);
    }
}
```

Источники конфигураций

```
public class DatabaseConfigurationProvider : ConfigurationProvider
{
    private readonly IRepositoryFactory _repositoryFactory;

    public DatabaseConfigurationProvider(IRepositoryFactory repositoryFactory)
    {
        _repositoryFactory = repositoryFactory;
    }

    public override void Load()
    {
        using var repository = _repositoryFactory.Create();
        Data = repository
            .GetTable<JobConfiguration>()
            .ToDictionary(v => v.Key, k => k.Value);
    }
}
```

Источники конфигураций

```
public class DatabaseConfigurationProvider : ConfigurationProvider
{
    private readonly IRepositoryFactory _repositoryFactory;

    public DatabaseConfigurationProvider(IRepositoryFactory repositoryFactory)
    {
        _repositoryFactory = repositoryFactory;
    }

    public override void Load()
    {
        using var repository = _repositoryFactory.Create();
        Data = repository
            .GetTable<JobConfiguration>()
            .ToDictionary(v => v.Key, k => k.Value);
    }
}
```

Источники конфигураций

Источники конфигураций

```
public class DatabaseConfigurationSource : IConfigurationSource
{
    private readonly IRepositoryFactory _repositoryFactory;

    public DatabaseConfigurationSource(IRepositoryFactory repositoryFactory)
    {
        _repositoryFactory = repositoryFactory;
    }

    public IConfigurationProvider Build()
    {
        return new DatabaseConfigurationProvider(_repositoryFactory);
    }
}
```

Источники конфигураций

```
public class DatabaseConfigurationSource : IConfigurationSource
{
    private readonly IRepositoryFactory _repositoryFactory;

    public DatabaseConfigurationSource(IRepositoryFactory repositoryFactory)
    {
        _repositoryFactory = repositoryFactory;
    }

    public IConfigurationProvider Build()
    {
        return new DatabaseConfigurationProvider(_repositoryFactory);
    }
}
```

Источники конфигураций

```
public static class ConfigurationBuilderExtensions
{
    public static IConfigurationBuilder AddDatabaseConfiguration(
        this IConfigurationBuilder builder)
    {
        var tempConfig = builder.Build();

        var connectionString = tempConfig.GetConnectionString("Database");

        var repositoryFactory = new RepositoryFactory(connectionString);

        return builder.Add(new DatabaseConfigurationSource(repositoryFactory));
    }
}
```

Источники конфигураций

```
public static class ConfigurationBuilderExtensions
{
    public static IConfigurationBuilder AddDatabaseConfiguration(
        this IConfigurationBuilder builder)
    {
        var tempConfig = builder.Build();

        var connectionString = tempConfig.GetConnectionString("Database");

        var repositoryFactory = new RepositoryFactory(connectionString);

        return builder.Add(new DatabaseConfigurationSource(repositoryFactory));
    }
}
```

Источники конфигураций

```
public static class ConfigurationBuilderExtensions
{
    public static IConfigurationBuilder AddDatabaseConfiguration(
        this IConfigurationBuilder builder)
    {
        var tempConfig = builder.Build();

        var connectionString = tempConfig.GetConnectionString("Database");

        var repositoryFactory = new RepositoryFactory(connectionString);

        return builder.Add(new DatabaseConfigurationSource(repositoryFactory));
    }
}
```

Применение решения

Применение решения

```
public ServiceConfigs()  
{  
    Host = Environment.GetEnvironmentVariable(@"HOST");  
}
```

Применение решения

```
public ServiceConfigs()  
{  
    Host = Environment.GetEnvironmentVariable(@"HOST");  
}
```

```
public ServiceConfigs(IConfiguration configuration)  
{  
    Host = configuration.GetValue<string>("Host");  
}
```


Применение решения

- Ручной маппинг
- Никакой структуры
- Отсутствие валидации
- Коммиты `launchSettings`
- Ограниченный набор источников значений

Применение решения

- Ручной маппинг
- Никакой структуры
- Отсутствие валидации
- Коммиты `launchSettings`
- ~~● Ограниченный набор источников значений~~

Применение решения

- Ручной маппинг
- Никакой структуры
- Отсутствие валидации
- ~~Коммиты launchSettings~~
- ~~Ограниченный набор источников значений~~

Применение решения

Применение решения

- Разделили параметры на группы

Применение решения

- Разделили параметры на группы
- Сформировали из них секции

Применение решения

- Разделили параметры на группы
- Сформировали из них секции
- Завели РОСО классы

Применение решения

Применение решения

```
var options = builder.Configuration  
    .GetSection("Article")  
    .Get<ArticleOptions>();
```

Применение решения

```
var options = builder.Configuration
    .GetSection("Article")
    .Get<ArticleOptions>();

services.AddSingleton(options);
```

Применение решения

```
var options = builder.Configuration
    .GetSection("Article")
    .Get<ArticleOptions>();
```

```
services.AddSingleton(options);
```

```
public Service(ArticleOptions options)
{
}
```

Options Pattern

Options Pattern

- Создание класса опций

Options Pattern

```
public sealed class ArticleOptions
{
    public const string Section = "Article";

    public string Title { get; set; }
    public string Author { get; set; }
}
```

Options Pattern

- Создание класса опций
- Конфигурация опций

Options Pattern

services

```
.AddOptions<ArticleOptions>()  
.Bind(configuration.GetSection(ArticleOptions.Section));
```



Options Pattern

- Создание класса опций
- Конфигурация опций
- Инъекция зависимостей

Options Pattern

- Создание класса опций
- Конфигурация опций
- Инъекция зависимостей
 - `IOptions<TOptions>`

Options Pattern

- Создание класса опций
- Конфигурация опций
- Инъекция зависимостей
 - `IOptions<TOptions>`
 - `IOptionsSnapshot<TOptions>`

Options Pattern

- Создание класса опций
- Конфигурация опций
- Инъекция зависимостей
 - `IOptions<TOptions>`
 - `IOptionsSnapshot<TOptions>`
 - `IOptionsMonitor<TOptions>`

Options Pattern

IOptions<TOptions>

IOptions<TOptions>

- Предоставляет только одно свойство - Value

IOptions<TOptions>

- Предоставляет только одно свойство - Value
- Используется для получения статических значений опций

IOptions<TOptions>

- Предоставляет только одно свойство - Value
- Используется для получения статических значений опций
- Регистрируется как singleton

IOptions<TOptions>

```
public class MyService
{
    private readonly ArticleOptions _options;

    public MyService(IOptions<ArticleOptions> options)
    {
        _options = options.Value;
    }
}
```

`IOptionsSnapshot<TOptions>`

IOptionsSnapshot<TOptions>

- Регистрируется как scoped

IOptionsSnapshot<TOptions>

- Регистрируется как `scoped`
- Работает с перезагружаемые конфигурации

IOptionsSnapshot<TOptions>

- Регистрируется как `scoped`
- Работает с перезагружаемые конфигурации
- Предоставляет возможность получать моментальные снимки (snapshot) значений опций

IOptionsSnapshot<TOptions>

- Регистрируется как `scoped`
- Работает с перезагружаемые конфигурации
- Предоставляет возможность получать моментальные снимки (snapshot) значений опций
- Поддерживает именованные параметры

IOptionsSnapshot<TOptions>

```
public class MyService
{
    private readonly ArticleOptions _options;

    public MyService(IOptionsSnapshot<ArticleOptions> options)
    {
        _options = options.Value;
    }
}
```


IOptionsSnapshot<TOptions>

```
public class MyService
{
    private readonly ArticleOptions _options;

    public MyService(IOptionsSnapshot<ArticleOptions> options)
    {
        _options = options.Value;
    }
}
```

`IOptionsSnapshot<TOptions>`

IOptionsSnapshot<TOptions>

```
public class Importer
{
    public const string BatchSize = 10;
    ...
    public void Import()
    {
        var entities = _provider.Get(BatchSize);
        ...
    }
    ...
}
```

IOptionsSnapshot<TOptions>

```
public class SyncOptions
{
    public int BatchSize { get; set; }
}
```

IOptionsSnapshot<TOptions>

```
{  
    "Import": {  
        "BatchSize": "10"  
    },  
    "Export": {  
        "BatchSize": "20"  
    }  
}
```

`IOptionsSnapshot<TOptions>`

IOptionsSnapshot<TOptions>

services

```
.AddOptions<SyncOptions>()  
.Bind(configuration.GetSection("Import"));
```

services

```
.AddOptions<SyncOptions>()  
.Bind(configuration.GetSection("Export"));
```

IOptionsSnapshot<TOptions>

services

```
.AddOptions<SyncOptions>("ImportOptions")
```


IOptionsSnapshot<TOptions>

services

```
.AddOptions<SyncOptions>("ImportOptions")  
.Bind(configuration.GetSection("Import"));
```

IOptionsSnapshot<TOptions>

services

```
.AddOptions<SyncOptions>("ImportOptions")  
.Bind(configuration.GetSection("Import"));
```

services

```
.AddOptions<SyncOptions>("ExportOptions")  
.Bind(configuration.GetSection("Export"));
```

IOptionsSnapshot<TOptions>

```
public class Importer
{
    private readonly int _batchSize;
    ...
    public Importer(IOptionsSnapshot<SyncOptions> options, ...)
    {
        _batchSize = options.Get("ImportOptions").BatchSize;
        ...
    }
}
```

`IOptionsMonitor<TOptions>`

IOptionsMonitor<TOptions>

- Регистрируется как singleton

IOptionsMonitor<TOptions>

- Регистрируется как singleton
- Работает с перезагружаемые конфигурации

IOptionsMonitor<TOptions>

- Регистрируется как singleton
- Работает с перезагружаемые конфигурации
- Предоставляет возможность отслеживать изменения в опциях в реальном времени

IOptionsMonitor<TOptions>

- Регистрируется как singleton
- Работает с перезагружаемые конфигурации
- Предоставляет возможность отслеживать изменения в опциях в реальном времени
- Поддерживает именованные параметры

IOptionsMonitor<TOptions>

```
public class MyService : IDisposable
{
    private ArticleOptions _options;
    private readonly IDisposable? _handler;

    public MyService(IOptionsMonitor<ArticleOptions> optionsMonitor)
    {
        _options = optionsMonitor.CurrentValue;
        _options = optionsMonitor.Get("OptionsMonitor");
        _handler = optionsMonitor.OnChange(newOptions => _options = newOptions);
    }

    public void Dispose()
    {
        _handler?.Dispose();
    }
}
```

IOptionsMonitor<TOptions>

```
public class MyService : IDisposable
{
    private ArticleOptions _options;
    private readonly IDisposable? _handler;

    public MyService(IOptionsMonitor<ArticleOptions> optionsMonitor)
    {
        _options = optionsMonitor.CurrentValue;
        _options = optionsMonitor.Get("OptionsMonitor");
        _handler = optionsMonitor.OnChange(newOptions => _options = newOptions);
    }

    public void Dispose()
    {
        _handler?.Dispose();
    }
}
```

IOptionsMonitor<TOptions>

```
public class MyService : IDisposable
{
    private ArticleOptions _options;
    private readonly IDisposable? _handler;

    public MyService(IOptionsMonitor<ArticleOptions> optionsMonitor)
    {
        _options = optionsMonitor.CurrentValue;
        _options = optionsMonitor.Get("OptionsMonitor");
        _handler = optionsMonitor.OnChange(newOptions => _options = newOptions);
    }

    public void Dispose()
    {
        _handler?.Dispose();
    }
}
```

IOptionsMonitor<TOptions>

```
public class MyService : IDisposable
{
    private ArticleOptions _options;
    private readonly IDisposable? _handler;

    public MyService(IOptionsMonitor<ArticleOptions> optionsMonitor)
    {
        _options = optionsMonitor.CurrentValue;
        _options = optionsMonitor.Get("OptionsMonitor");
        _handler = optionsMonitor.OnChange(newOptions => _options = newOptions);
    }

    public void Dispose()
    {
        _handler?.Dispose();
    }
}
```

IOptionsMonitor<TOptions>

```
public class MyService : IDisposable
{
    private ArticleOptions _options;
    private readonly IDisposable? _handler;

    public MyService(IOptionsMonitor<ArticleOptions> optionsMonitor)
    {
        _options = optionsMonitor.CurrentValue;
        _options = optionsMonitor.Get("OptionsMonitor");
        _handler = optionsMonitor.OnChange(newOptions => _options = newOptions);
    }

    public void Dispose()
    {
        _handler?.Dispose();
    }
}
```

IOptionsMonitor<TOptions>

```
public class MyService
{
    private readonly IOptionsMonitor<ArticleOptions> _optionsMonitor;

    public MyService(IOptionsMonitor<ArticleOptions> optionsMonitor)
    {
        _optionsMonitor = optionsMonitor;
    }

    public string GetAuthor()
    {
        return _optionsMonitor.CurrentValue.Author;
    }
}
```

Options Pattern

Применение решения

- Ручной маппинг
- Никакой структуры
- Отсутствие валидации
- ~~Коммиты launchSettings~~
- ~~Ограниченный набор источников значений~~

Применение решения

- ~~Ручной маппинг~~
- Никакой структуры
- Отсутствие валидации
- ~~Коммиты launchSettings~~
- ~~Ограниченный набор источников значений~~

Применение решения

- ~~Ручной маппинг~~
- ~~Никакой структуры~~
- Отсутствие валидации
- ~~Коммиты launchSettings~~
- ~~Ограниченный набор источников значений~~

Валидация

Валидация

```
public sealed class ArticleOptions
{
    public const string Section = "Article";

    [MaxLength(200)]
    public string Title { get; set; }

    [Required]
    public string Author { get; set; }
}
```

Валидация

services

```
.AddOptions<ArticleOptions>()  
.Bind(builder.Configuration.GetSection(ArticleOptions.Section))  
.ValidateDataAnnotations();
```

Валидация

```
public interface IValidateOptions<TOptions> where TOptions : class
{
    ValidateOptionsResult Validate(string? name, TOptions options);
}
```

Валидация

```
public interface IValidateOptions<TOptions> where TOptions : class
{
    ValidateOptionsResult Validate(string? name, TOptions options);
}

services.AddSingleton<IValidateOptions<ArticleOptions>, ArticleValidation>();
```

Валидация

```
public class ArticleOptionsValidator : AbstractValidator<ArticleOptions>,
    IValidateOptions<ArticleOptions>
{
    public ValidateOptionsResult Validate(string? name, ArticleOptions options)
    {
        var result = Validate(options);
        ...
    }
}
```


Валидация

```
public class ArticleOptionsValidator : AbstractValidator<ArticleOptions>,  
    IValidateOptions<ArticleOptions>  
{  
    public ValidateOptionsResult Validate(string? name, ArticleOptions options)  
    {  
        var result = Validate(options);  
        ...  
    }  
}
```

Валидация

```
public class ArticleOptionsValidator : AbstractValidator<ArticleOptions>,
    IValidateOptions<ArticleOptions>
{
    public ValidateOptionsResult Validate(string? name, ArticleOptions options)
    {
        var result = Validate(options);
        ...
    }
}
```

Валидация

```
public class ArticleOptionsValidator : AbstractValidator<ArticleOptions>,
    IValidateOptions<ArticleOptions>
{
    public ValidateOptionsResult Validate(string? name, ArticleOptions options)
    {
        var result = Validate(options);
        ...
    }
}
```

Валидация

```
public class FluentValidationOptions<TOptions>
    : IValidateOptions<TOptions> where TOptions : class
{
    ...
    var validator = scope.ServiceProvider
        .GetRequiredService<IValidator<TOptions>>();

    var results = validator.Validate(options);
    ...
}
```

Валидация

```
public class FluentValidationOptions<TOptions>
    : IValidateOptions<TOptions> where TOptions : class
{
    ...
    var validator = scope.ServiceProvider
        .GetRequiredService<IValidator<TOptions>>();

    var results = validator.Validate(options);
    ...
}
```

Валидация

```
public class FluentValidationOptions<TOptions>
    : IValidateOptions<TOptions> where TOptions : class
{
    ...
    var validator = scope.ServiceProvider
        .GetRequiredService<IValidator<TOptions>>();

    var results = validator.Validate(options);
    ...
}
```

Валидация

```
public class FluentValidationOptions<TOptions>
    : IValidateOptions<TOptions> where TOptions : class
{
    ...
    var validator = scope.ServiceProvider
        .GetRequiredService<IValidator<TOptions>>();

    var results = validator.Validate(options);
    ...
}
```

Валидация

```
public static OptionsBuilder<TOptions> ValidateFluentValidation<TOptions>(
    this OptionsBuilder<TOptions> optionsBuilder) where TOptions : class
{
    optionsBuilder.Services.AddSingleton<IValidateOptions<TOptions>>(
        provider => new FluentValidationOptions<TOptions>(
            optionsBuilder.Name, provider));
    return optionsBuilder;
}
```


Валидация

```
public static OptionsBuilder<TOptions> ValidateFluentValidation<TOptions>(
    this OptionsBuilder<TOptions> optionsBuilder) where TOptions : class
{
    optionsBuilder.Services.AddSingleton<IValidateOptions<TOptions>>(
        provider => new FluentValidationOptions<TOptions>(
            optionsBuilder.Name, provider));
    return optionsBuilder;
}
```

services

```
.AddOptions<ArticleOptions>()
.Bind(builder.Configuration.GetSection(ArticleOptions.Section))
.ValidateFluentValidation();
```

Валидация

services

```
.AddOptions<ArticleOptions>()  
.Bind(configuration.GetSection(ArticleOptions.Section))  
.Validate(config => config.Title is not null, "Ошибка :(");
```

Валидация

services

```
.AddOptions<ArticleOptions>()  
.Bind(builder.Configuration.GetSection(ArticleOptions.Section))
```

Валидация

services

```
.AddOptions<ArticleOptions>()  
.Bind(builder.Configuration.GetSection(ArticleOptions.Section))  
.ValidateOnStart();
```

Валидация

services

```
.AddOptions<ArticleOptions>()  
.Bind(builder.Configuration.GetSection(ArticleOptions.Section))  
.ValidateOnStart();
```

services

```
.AddOptionsWithValidateOnStart<ArticleOptions>()  
.Bind(builder.Configuration.GetSection(ArticleOptions.Section));
```

Применение решения

- ~~Ручной маппинг~~
- ~~Никакой структуры~~
- Отсутствие валидации
- ~~Коммиты launchSettings~~
- ~~Ограниченный набор источников значений~~

Применение решения

- Ручной маппинг
- Никакой структуры
- Отсутствие валидации
- Коммиты `launchSettings`
- Ограниченный набор источников значений

Пост конфигурация

Пост конфигурация

services

```
.AddOptions<ArticleOptions>()  
.Bind(builder.Configuration.GetSection(ArticleOptions.Section))  
.PostConfigure(options =>  
{  
    if (options.Author.EndsWith(" "))  
    {  
        options.Author = options.Author.TrimEnd();  
    }  
});
```

Пост конфигурация

services

```
.AddOptions<ArticleOptions>()  
.Bind(builder.Configuration.GetSection(ArticleOptions.Section))  
.PostConfigure<Normalizer>((options, normalizer) =>  
{  
    options.Author = normalizer.Normalize(options.Author);  
});
```

Пост конфигурация

```
services
    .PostConfigureAll<SyncOptions>(options =>
    {
        if (options.BatchSize < 15)
        {
            options.BatchSize = 15;
        }
    });
```

Пост конфигурация

```
public interface IPostConfigureOptions<in TOptions> where TOptions : class
{
    void Validate(string? name, TOptions options);
}
```

Пост конфигурация

```
public interface IPostConfigureOptions<in TOptions> where TOptions : class
{
    void Validate(string? name, TOptions options);
}
```

```
services.AddSingleton<
    IPostConfigureOptions<ArticleOptions>,
    ArticleOptionsPostConfigure>();
```

Что мы получили

Что мы получили

- Структурированные классы опций

Что мы получили

- Структурированные классы опций
- Автоматический биндинг

Что мы получили

- Структурированные классы опций
- Автоматический биндинг
- Options pattern

Что мы получили

- Структурированные классы опций
- Автоматический биндинг
- Options pattern
- Валидация

Что мы получили

- Структурированные классы опций
- Автоматический биндинг
- Options pattern
- Валидация
- Разные провайдеры

Что мы получили

- Структурированные классы опций
- Автоматический биндинг
- Options pattern
- Валидация
- Разные провайдеры
- Пост-конфигурация

Что мы получили

```
[Test]
public void DoWork_ShouldReturnTrue_ThenWorkIsValid()
{
    var config = Options.Create(new ServiceConfigs
    {
        Period = 10
    });
    var config = new ServiceConfigs();

    var service = new Service(config);
    // Act
    // Assert
}
```

И чем же это хорошо?

И чем же это хорошо?

- Это не велосипед :)

И чем же это хорошо?

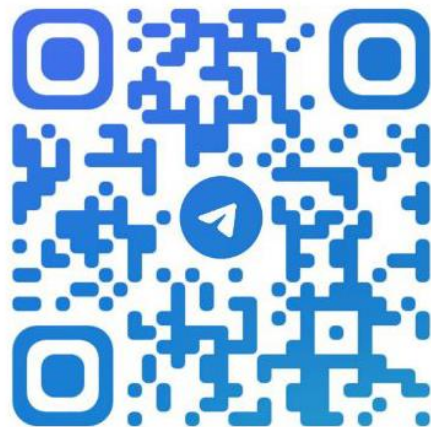
- Это не велосипед :)
- Поддержка и дальнейшее развитие

И чем же это хорошо?

- Это не велосипед :)
- Поддержка и дальнейшее развитие
- Переход занимает буквально 1 день :)

2ГИС

Спасибо
за внимание!



@ANDREW_REAGUZOV