Садимся на два стула одновременно

изолированные функциональные тесты Asp.Net Core WebApi

Кугушев Александр

Lead Software Engineer в EPAM

kugushew@gmail.com

www.linkedin.com/in/kugushev/

github.com/AleksandrKugushev

Подкаст DotNet & More

https://dotnetmore.ru

VK: https://vk.com/dotnetmore

Rss: https://dotnetmore.ru/feed/podcast/

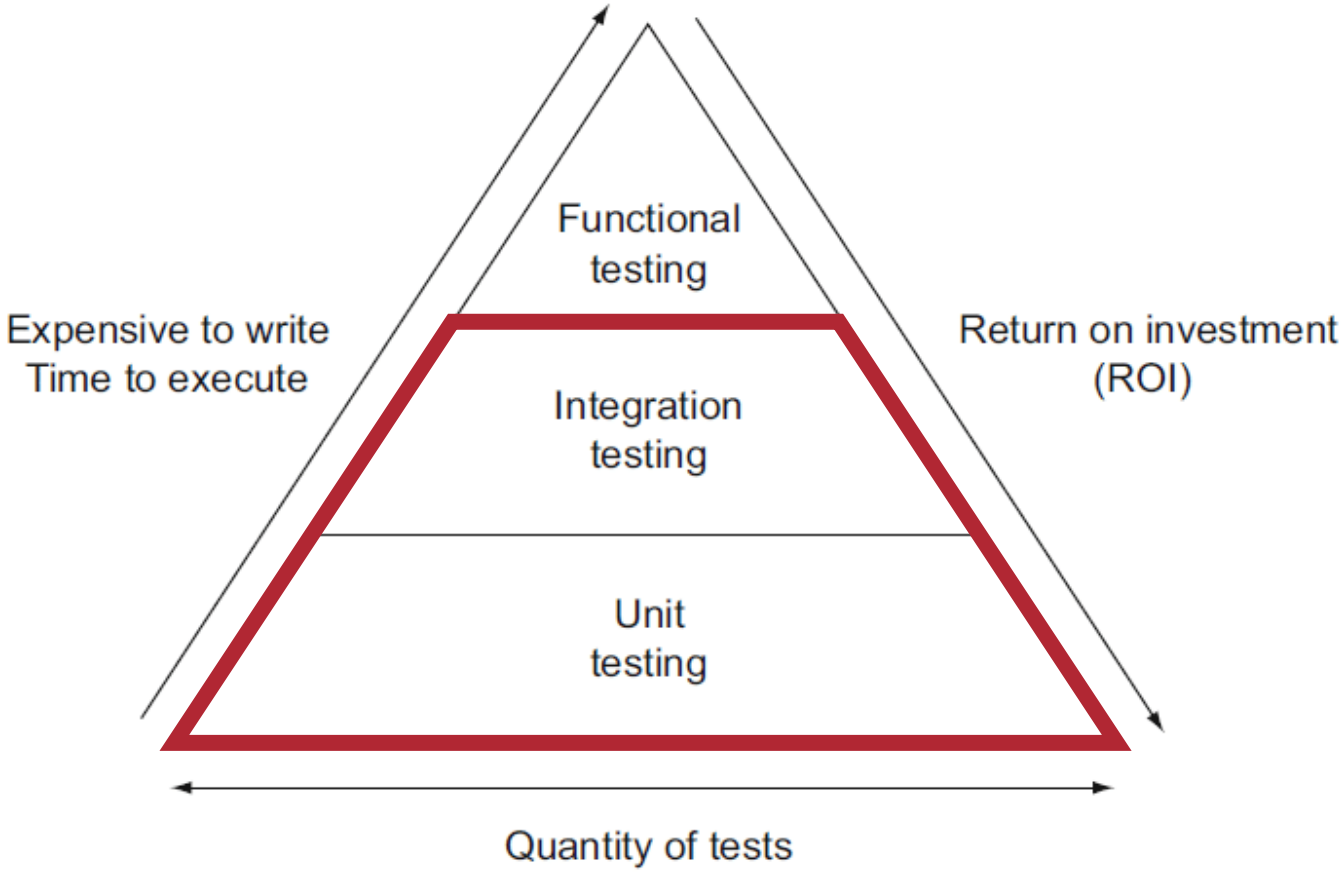
SoundCloud: https://soundcloud.com/dotnetmore

Twitter: https://twitter.com/dotnetmore

Telegram channel: https://t.me/dotnetmore

Telegram chat: https://t.me/dotnetmore_chat

Виды тестирования



Виды тестирования

	Unit Testing	Integration Testing	Functional Testing
Цель			
Сложность			
Методология			
Изоляция			
Переносимость			

Виды тестирования

	Unit Testing	Integration Testing	Functional Testing
Цель	Модуль		
Сложность			
Методология			
Изоляция			
Переносимость			

Виды тестирования

	Unit Testing	Integration Testing	Functional Testing
Цель	Модуль	Интеграция	
Сложность			
Методология			
Изоляция			
Переносимость			

Виды тестирования

	Unit Testing	Integration Testing	Functional Testing
Цель	Модуль	Интеграция	Функциональность
Сложность			
Методология			
Изоляция			
Переносимость			

Виды тестирования

	Unit Testing	Integration Testing	Functional Testing
Цель	Модуль	Интеграция	Функциональность
Сложность			
Методология			
Изоляция			
Переносимость			

Виды тестирования

	Unit Testing	Integration Testing	Functional Testing
Цель	Модуль	Интеграция	Функциональность
Сложность		?	
Методология			
Изоляция			
Переносимость			

Виды тестирования

	Unit Testing	Integration Testing	Functional Testing
Цель	Модуль	Интеграция	Функциональность
Сложность		?	?
Методология			
Изоляция			
Переносимость			

Виды тестирования

	Unit Testing	Integration Testing	Functional Testing
Цель	Модуль	Интеграция	Функциональность
Сложность		?	?
Методология	White box		
Изоляция			
Переносимость			

Виды тестирования

	Unit Testing	Integration Testing	Functional Testing
Цель	Модуль	Интеграция	Функциональность
Сложность		?	?
Методология	White box	White/Black box	
Изоляция			
Переносимость			

Виды тестирования

	Unit Testing	Integration Testing	Functional Testing
Цель	Модуль	Интеграция	Функциональность
Сложность		?	?
Методология	White box	White/Black box	Black box
Изоляция			
Переносимость			

Виды тестирования

	Unit Testing	Integration Testing	Functional Testing
Цель	Модуль	Интеграция	Функциональность
Сложность		?	?
Методология	White box	White/Black box	Black box
Изоляция			
Переносимость			

Виды тестирования

	Unit Testing	Integration Testing	Functional Testing
Цель	Модуль	Интеграция	Функциональность
Сложность			
Методология	White box	White/Black box	Black box
Изоляция			
Переносимость			

Виды тестирования

	Unit Testing	Integration Testing	Functional Testing
Цель	Модуль	Интеграция	Функциональность
Сложность			
Методология	White box	White/Black box	Black box
Изоляция			
Переносимость			

Виды тестирования

	Unit Testing	Integration Testing	Functional Testing
Цель	Модуль	Интеграция	Функциональность
Сложность			
Методология	White box	White/Black box	Black box
Изоляция			
Переносимость			

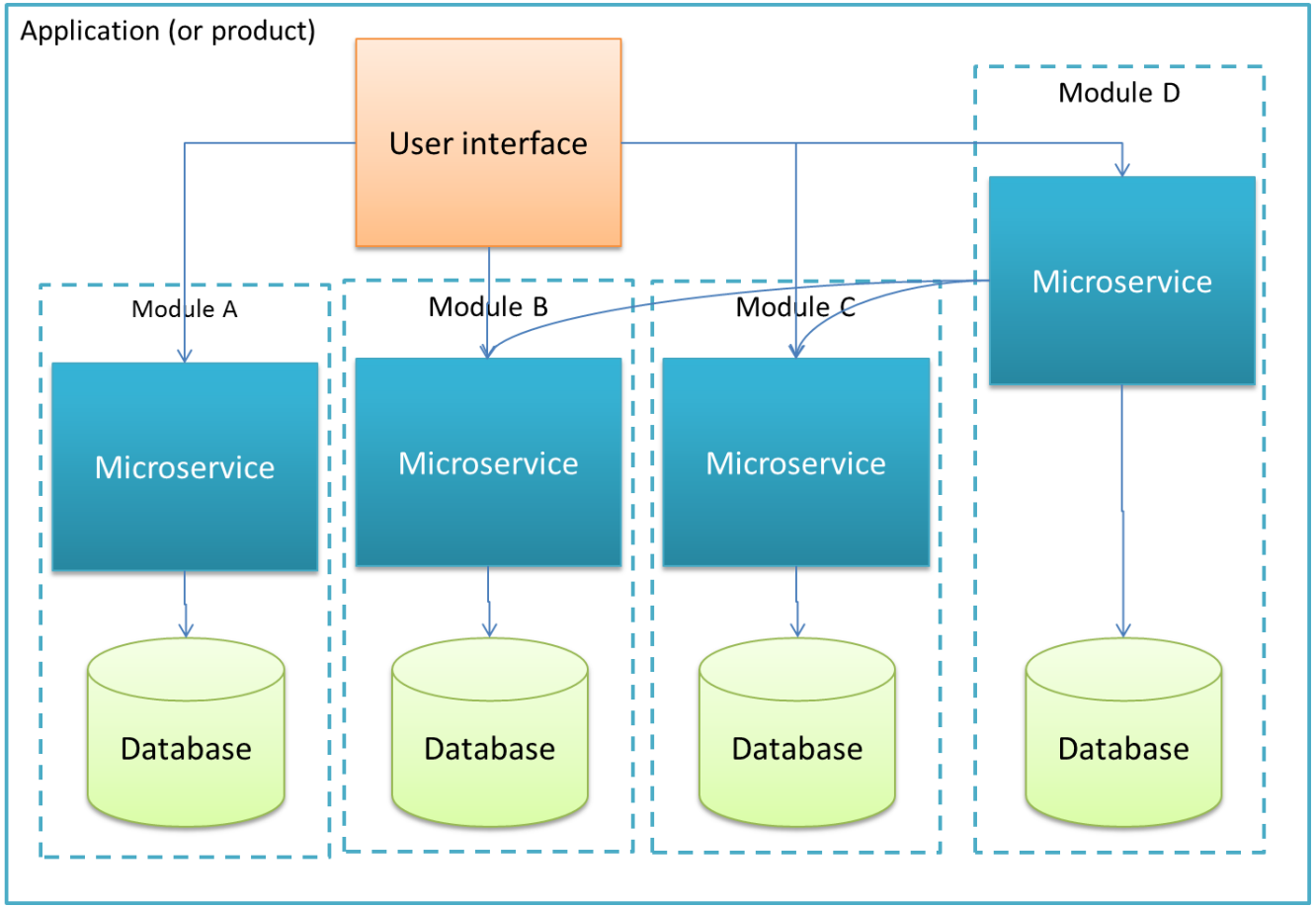
Виды тестирования

	Unit Testing	Integration Testing	Functional Testing
Цель	Модуль	Интеграция	Функциональность
Сложность			
Методология	White box	White/Black box	Black box
Изоляция			
Переносимость			

Виды тестирования

	Unit Testing	Integration Testing	Functional Testing
Цель	Модуль	Интеграция	Функциональность
Сложность			
Методология	White box	White/Black box	Black box
Изоляция			
Переносимость			

Тестирование и микросервисы



Микросервис

- Модуль
- Ответственен только за часть бизнес логики
- Изолирован от других сервисов

- Более чем просто метод
- Бизнес логика может быть достаточно сложна
- Инфраструктурные зависимости: Auth, SSL, и т.д.

Виды тестирования

	Unit Testing	Integration Testing	Functional Testing
Цель	Модуль		
Сложность			
Методология	White box		
Изоляция			
Переносимость			

- Сильно привязяны к реализации
- Отчет не показать BA, Product Owner и т.д.
- Связь с бизнес требованиями не очевидна

Виды тестирования

	Unit Testing	Integration Testing	Functional Testing
		Интеграция	Функциональность
		?	?
		White/Black box	Black box
			?
			?

- Бизнес требования, обычно, не затрагиваются
- Низкая изолированность
- Требуют времени для локального развертывания

Виды тестирования

	Unit Testing	Integration Testing	Functional Testing
Цель			Функциональность
Сложность			?
Методология			Black box
Изоляция			?
Переносимость			?

- Не допускают вмешательства в код
- Требуют полного разворачивания всей сисемы
- Требуют много времени для локального развертывания

Виды тестирования

	Unit Testing	Integration Testing	Functional Testing
Цель	Модуль	Интеграция	Функциональность
Сложность		?	?
Методология	White box	White/Black box	Black box
Изоляция			?
Переносимость			?

Виды тестирования

	Unit Testing	Integration Testing	Functional Testing
Цель	Модуль	Интеграция	Функциональность
Сложность			
Методология	White box	White/Black box	Black box
Изоляция			
Переносимость			

Виды тестирования

	Unit Testing	Integration Testing	Functional Testing
Цель	Модуль	Интеграция	Функциональность
Сложность		?	?
Методология	White box	White/Black box	Black box
Изоляция			?
Переносимость			?

Виды тестирования

	Unit Testing	Integration Testing	Functional Testing
Цель	Модуль	Интеграция	Функциональность
Сложность		?	?
Методология	White box	White/Black box	Black box
Изоляция			?
Переносимость			?

Виды тестирования

	Unit Testing	Integration Testing	Functional Testing
Цель	Модуль	Интеграция	Функциональность
Сложность		?	?
Методология	White box	White/Black box	Black box
Изоляция			?
Переносимость			?

Виды тестирования

	Unit Testing	Integration Testing	Functional Testing
Цель	Модуль	Интеграция	Функциональность
Сложность		?	?
Методология	White box	White/Black box	Black box
Изоляция			?
Переносимость			?

Решаемые проблемы

- Дополнение к git blame
  - Всегда добавляйте номер таски к названию теста
- Проще покрыть legacy код
- Быстрый on-boarding новичков
- Вовлечение BA и Product Owner в разработку
  - Проще «продать» задачу на написание тестов
  - С разработчиков снимается часть ответственности

А может отдать это все Auto QA?

Можно, но тогда мы потеряем:

- Изолированность
- Переносимость
- Простоту разработки

Функциональные тесты WebApi

- Фокус на бизнес требованиях
- Достаточно просты
- Допускают white box, но чуть-чуть
- Максимально независимы от окружения
- Запускаются так же просто, как и модульные тесты

Mutants Catalogue

- GET api/combat?attacker={attacker}&defender={defender}
- GET api/mutants/{name}
- GET api/combat/epic?attacker={attacker}&defender={defender}

Combat

GET /api/Combat

GET /api/Combat/epic

Mutants

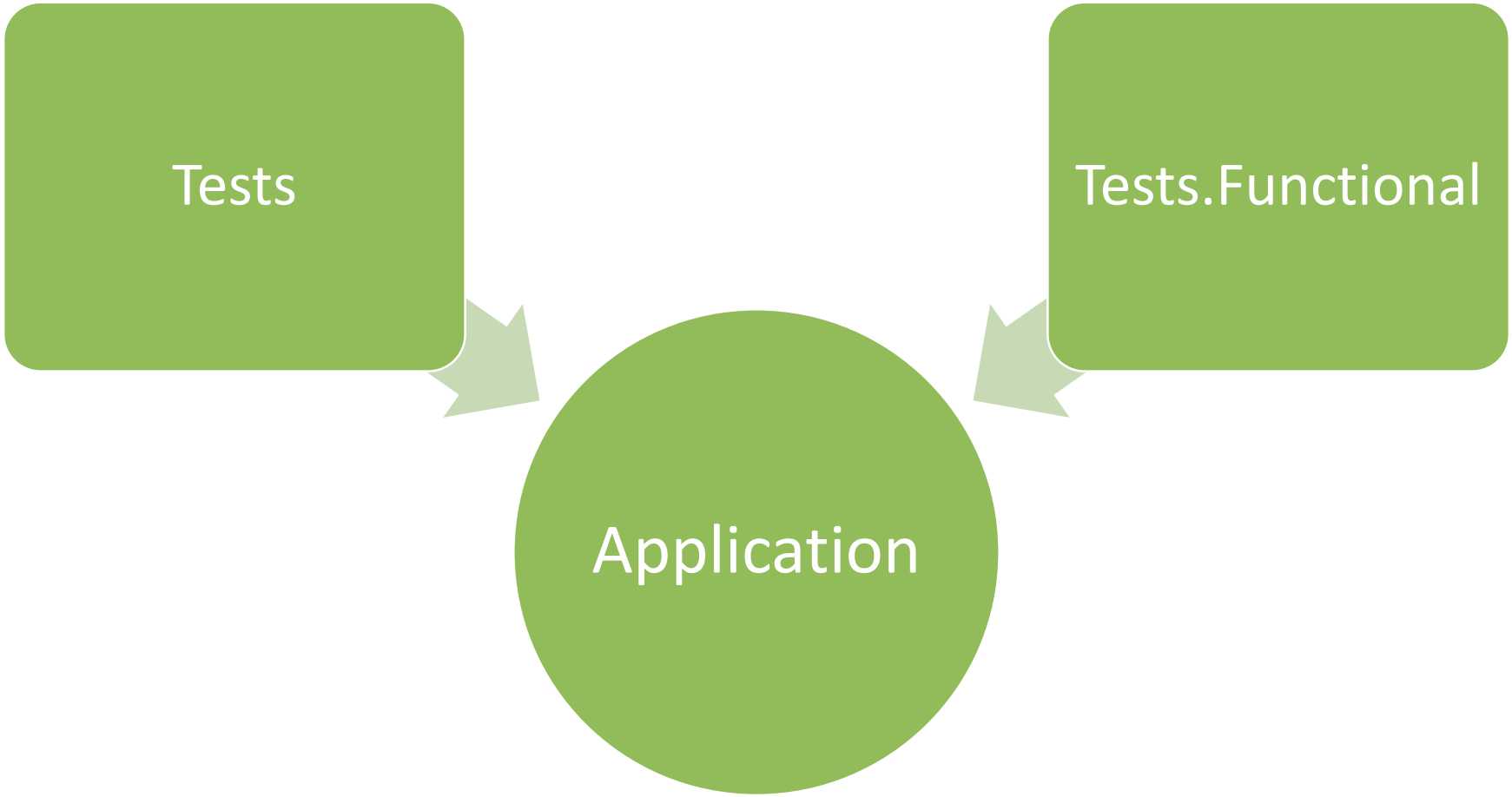
GET /api/Mutants/{name}

Models

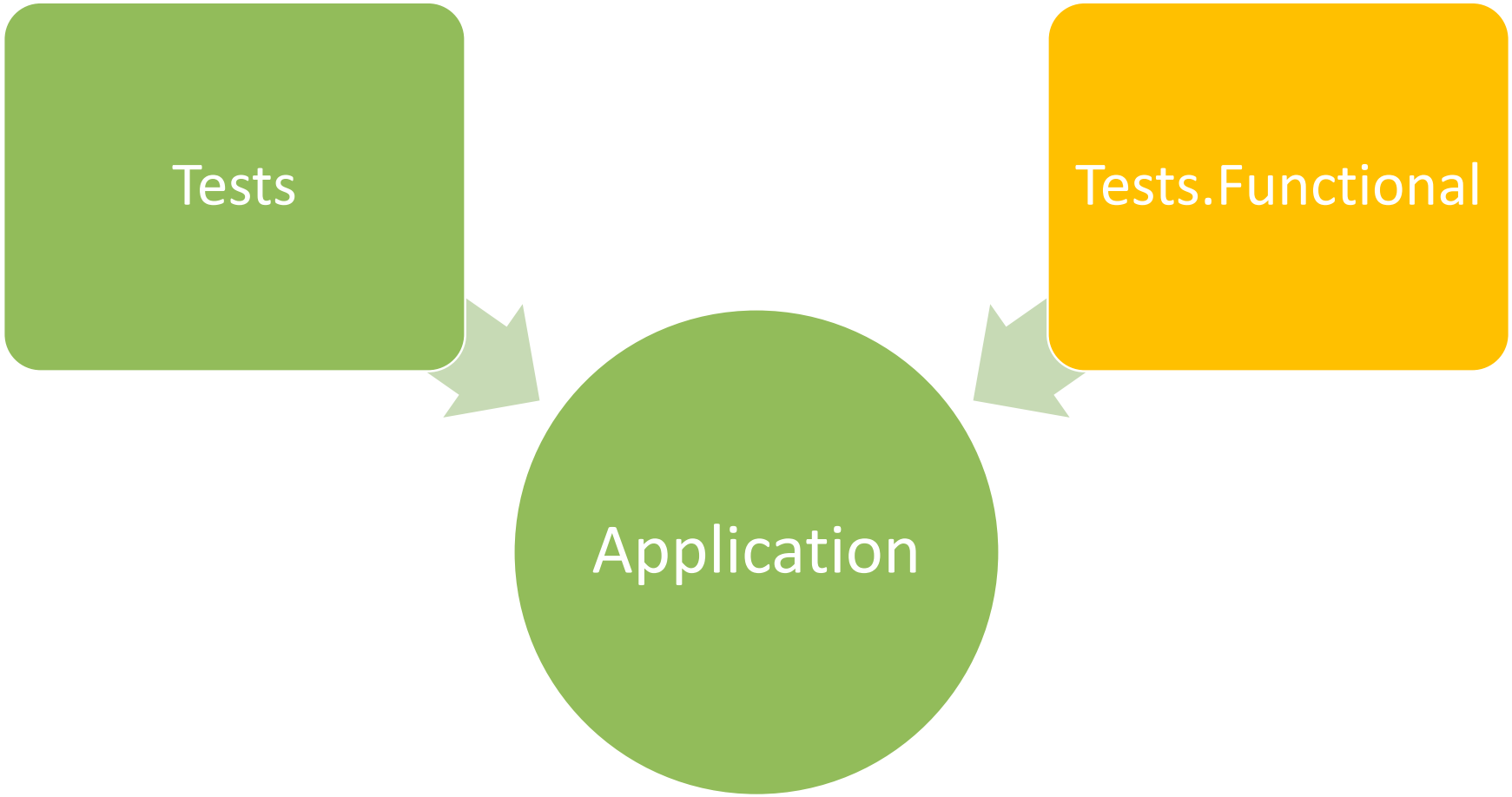
```
CombatResult ⌄ {
    winner              string
    victoryPhrase       string
    copyright           string
}
```

```
Mutant ⌄ {
    name                string
    realName            string
    superpower          string
}
```

Структура проекта



Структура проекта



Создаем тестовый проект

```
<Project Sdk="Microsoft.NET.Sdk.Web">
<PropertyGroup><TargetFramework>netcoreapp2.2</TargetFramework</PropertyGroup>
  <ItemGroup>
    <PackageReference Include="Microsoft.AspNetCore.App"/>
    <PackageReference Include="Microsoft.AspNetCore.Mvc.Testing"  Version="2.2.0"/>
    <PackageReference Include="Microsoft.NET.Test.Sdk" Version="15.9.0" />
    <PackageReference Include="xunit" Version="2.4.0" />
    <PackageReference Include="xunit.runner.visualstudio" Version="2.4.0" />
  </ItemGroup>
</Project>
```

Создаем тестовый проект

```
<Project Sdk="Microsoft.NET.Sdk.Web">
<PropertyGroup><TargetFramework>netcoreapp2.2</TargetFramework</PropertyGroup>
  <ItemGroup>
    <PackageReference Include="Microsoft.AspNetCore.App"/>
    <PackageReference Include="Microsoft.AspNetCore.Mvc.Testing"  Version="2.2.0"/>
    <PackageReference Include="Microsoft.NET.Test.Sdk" Version="15.9.0" />
    <PackageReference Include="xunit" Version="2.4.0" />
    <PackageReference Include="xunit.runner.visualstudio" Version="2.4.0" />
  </ItemGroup>
</Project>
```

WebApplicationFactory

```
public class CombatFeature : IClassFixture<WebApplicationFactory<Startup>>
{
  private readonly WebApplicationFactory<Startup> factory;

  public CombatFeature(WebApplicationFactory<Startup> factory)
  {
    this.factory = factory;
  }
…
```

#1: Wolverine vs Magneto

```
[Fact]
public async Task Combat_MagnetoVsWolverine_MagnetoWins()
{
  // arrange
  var client = factory.CreateClient();
  // act
  var response = await client.
              GetAsync("api/combat?attacker=Magneto&defender=Wolverine");
  // assert
  response.EnsureSuccessStatusCode();
  var result = await response.Content.ReadAsAsync<CombatResult>();
  Assert.Equal("Magneto", result.Winner);
}
```

#1: Wolverine vs Magneto

```
[Fact]
public async Task Combat_MagnetoVsWolverine_MagnetoWins()
{
  // arrange
  var client = factory.CreateClient();
  // act
  var response = await client.
              GetAsync("api/combat?attacker=Magneto&defender=Wolverine");
  // assert
  response.EnsureSuccessStatusCode();
  var result = await response.Content.ReadAsAsync<CombatResult>();
  Assert.Equal("Magneto", result.Winner);
}
```

#1: Wolverine vs Magneto

```
[Fact]
public async Task Combat_MagnetoVsWolverine_MagnetoWins()
{
  // arrange
  var client = factory.CreateClient();
  // act
  var response = await client.
              GetAsync("api/combat?attacker=Magneto&defender=Wolverine");
  // assert
  response.EnsureSuccessStatusCode();
  var result = await response.Content.ReadAsAsync<CombatResult>();
  Assert.Equal("Magneto", result.Winner);
}
```

Test#1: Wolverine vs Magneto

```
[Fact]
public async Task Combat_MagnetoVsWolverine_MagnetoWins()
{
  // arrange
  var client = factory.CreateClient();
  // act
  var response = await client.
              GetAsync("api/combat?attacker=Magneto&defender=Wolverine");
  // assert
  response.EnsureSuccessStatusCode();
  var result = await response.Content.ReadAsAsync<CombatResult>();
  Assert.Equal("Magneto", result.Winner);
}
```

Request URL

```
http://localhost:58381/api/Combat?attacker=Wolverine&defender=Magne
```

Server response

Code	Details
200	Response body

```
{
  "winner": "Magneto",
  "victoryPhrase": null,
  "copyright": "Aleksandr Kugushev 2019"
}
```

Copyright

- appSettings.json
  - Copyright: Aleksandr Kugushev
- appSettings.Development.json
  - CopyrightYear: 2019

Наследник WebApplicationFactory

```
public class TestWebApplicationFactory : WebApplicationFactory<Startup> {
  protected override IWebHostBuilder CreateWebHostBuilder() {
    return WebHost.CreateDefaultBuilder().UseStartup<Startup>();
  }
  protected override void ConfigureWebHost(IWebHostBuilder builder){
    var config = new Dictionary<string, string>{
                    ["Domain:CopyrightYear"] = "42"
                };
    builder.ConfigureAppConfiguration(
                b => b.AddInMemoryCollection(config));
    base.ConfigureWebHost(builder);
  }
}
```

Наследник WebApplicationFactory

```
public class TestWebApplicationFactory : WebApplicationFactory<Startup> {
  protected override IWebHostBuilder CreateWebHostBuilder() {
    return WebHost.CreateDefaultBuilder().UseStartup<Startup>();
  }
  protected override void ConfigureWebHost(IWebHostBuilder builder){
    var config = new Dictionary<string, string>{
                    ["Domain:CopyrightYear"] = "42"
                };
    builder.ConfigureAppConfiguration(
                b => b.AddInMemoryCollection(config));
    base.ConfigureWebHost(builder);
  }
}
```

Наследник WebApplicationFactory

```
public class TestWebApplicationFactory : WebApplicationFactory<Startup> {
  protected override IWebHostBuilder CreateWebHostBuilder() {
    return WebHost.CreateDefaultBuilder().UseStartup<Startup>();
  }
  protected override void ConfigureWebHost(IWebHostBuilder builder){
    var config = new Dictionary<string, string>{
                    ["Domain:CopyrightYear"] = "42"
                 };
    builder.ConfigureAppConfiguration(
                 b => b.AddInMemoryCollection(config));
    base.ConfigureWebHost(builder);
  }
}
```

Наследник WebApplicationFactory

```
public class TestWebApplicationFactory : WebApplicationFactory<Startup> {
  protected override IWebHostBuilder CreateWebHostBuilder() {
    return WebHost.CreateDefaultBuilder().UseStartup<Startup>();
  }
  protected override void ConfigureWebHost(IWebHostBuilder builder){
    var config = new Dictionary<string, string>{
                    ["Domain:CopyrightYear"] = "42"
                };
    builder.ConfigureAppConfiguration(
                b => b.AddInMemoryCollection(config));
    base.ConfigureWebHost(builder);
  }
}
```

Наследник WebApplicationFactory

```
public class TestWebApplicationFactory : WebApplicationFactory<Startup> {
  protected override IWebHostBuilder CreateWebHostBuilder() {
    return WebHost.CreateDefaultBuilder().UseStartup<Startup>();
  }
  protected override void ConfigureWebHost(IWebHostBuilder builder){
    var config = new Dictionary<string, string>{
                    ["Domain:CopyrightYear"] = "42"
                 };
    builder.ConfigureAppConfiguration(
                 b => b.AddInMemoryCollection(config));
    base.ConfigureWebHost(builder);
  }
}
```

Наследник WebApplicationFactory

```
public class TestWebApplicationFactory : WebApplicationFactory<Startup> {
  protected override IWebHostBuilder CreateWebHostBuilder() {
    return WebHost.CreateDefaultBuilder().UseStartup<Startup>();
  }
  protected override void ConfigureWebHost(IWebHostBuilder builder){
    var config = new Dictionary<string, string>{
                    ["Domain:CopyrightYear"] = "42"
                };
    builder.ConfigureAppConfiguration(
                b => b.AddInMemoryCollection(config));
    base.ConfigureWebHost(builder);
  }
}
```

Test#2: Returns Copyright

```
[Fact]
public async Task Combat_AnyMutant_ReturnsCopiright(){
  // arrange
  var client = factory.CreateClient();
  // act
  var response = await client
      .GetAsync("api/combat?attacker=Magneto&defender=Xavier");
  // assert
  response.EnsureSuccessStatusCode();
  var result = await response.Content.ReadAsAsync<CombatResult>();
  Assert.Equal("Aleksandr Kugushev 42", result.Copyright);
}
```

Test#2: Returns Copyright

```
[Fact]
public async Task Combat_AnyMutant_ReturnsCopiright(){
  // arrange
  var client = factory.CreateClient();
  // act
  var response = await client
      .GetAsync("api/combat?attacker=Magneto&defender=Xavier");
  // assert
  response.EnsureSuccessStatusCode();
  var result = await response.Content.ReadAsAsync<CombatResult>();
  Assert.Equal("Aleksandr Kugushev 42", result.Copyright);
}
```

Middleware and Services

```
public class Startup {
  public void ConfigureServices(IServiceCollection services){
            services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version_2_2);
            services.AddSwaggerGen(…)
            services.AddAuth();
            services.AddDomain();
            services.AddDal();
            services.AddInfrastructure();
        }
  public void Configure(IApplicationBuilder app, IHostingEnvironment env){
            app.UseSwagger();
            app.UseAuthentication();
            app.UseHttpsRedirection();
            app.UseMvc();
        }
```

Middleware and Services

```
public class Startup {
  public void ConfigureServices(IServiceCollection services){
            services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version_2_2);
            services.AddSwaggerGen(…)
            services.AddAuth();
            services.AddDomain();
            services.AddDal();
            services.AddInfrastructure();
        }
  public void Configure(IApplicationBuilder app, IHostingEnvironment env){
            app.UseSwagger();
            app.UseAuthentication();
            app.UseHttpsRedirection();
            app.UseMvc();
        }
```

Middleware and Services

```
public class Startup {
  public void ConfigureServices(IServiceCollection services){
            services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version_2_2);
            ConfigureUtilityServices(services);
            services.AddDomain();
            services.AddDal();
            services.AddInfrastructure();
        }
  protected virtual void ConfigureUtilityServices(IServiceCollection services)
  public void Configure(IApplicationBuilder app, IHostingEnvironment env){
            ConfigureUtilityMiddlewares(app, env);
            app.UseMvc();
        }
  protected virtual void ConfigureUtilityMiddlewares(IApplicationBuilder app, IHostingEnvironment
env)
```

Middleware and Services

```
public class Startup {
  public void ConfigureServices(IServiceCollection services){
            services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version_2_2);
            ConfigureUtilityServices(services);
            services.AddDomain();
            services.AddDal();
            services.AddInfrastructure();
        }
  protected virtual void ConfigureUtilityServices(IServiceCollection services)
  public void Configure(IApplicationBuilder app, IHostingEnvironment env){
            ConfigureUtilityMiddlewares(app, env);
            app.UseMvc();
        }
  protected virtual void ConfigureUtilityMiddlewares(IApplicationBuilder app, IHostingEnvironment
env)
```

TestStartup

```
public class TestStartup : Startup {

  protected override void ConfigureUtilityMiddlewares(
        IApplicationBuilder app, IHostingEnvironment env){
    base.ConfigureUtilityMiddlewares(app, env);
  }

  protected override void ConfigureUtilityServices(
        IServiceCollection services){
    base.ConfigureUtilityServices(services);
  }
}
```

TestStartup

```
public class TestStartup : Startup {

  protected override void ConfigureUtilityMiddlewares(
        IApplicationBuilder app, IHostingEnvironment env){

  }

  protected override void ConfigureUtilityServices(
        IServiceCollection services){

  }
}
```

WebApplicationFactory<TestStartup>

```
public class TestWebApplicationFactory : WebApplicationFactory<TestStartup> {
  protected override IWebHostBuilder CreateWebHostBuilder() {
    return WebHost.CreateDefaultBuilder().UseStartup<Startup>();
  }
  protected override void ConfigureWebHost(IWebHostBuilder builder){
    var config = new Dictionary<string, string>{
                    ["Domain:CopyrightYear"] = "42"
                };
    builder.ConfigureAppConfiguration(
                b => b.AddInMemoryCollection(config));
    base.ConfigureWebHost(builder);
  }
}
```

WebApplicationFactory<TestStartup>

```
public class TestWebApplicationFactory : WebApplicationFactory<TestStartup> {
  protected override IWebHostBuilder CreateWebHostBuilder() {
    return WebHost.CreateDefaultBuilder().UseStartup<Startup>();
  }
  protected override void ConfigureWebHost(IWebHostBuilder builder){
    var config = new Dictionary<string, string>{
                    ["Domain:CopyrightYear"] = "42"
                 };
    builder.ConfigureAppConfiguration(
                 b => b.AddInMemoryCollection(config));
    base.ConfigureWebHost(builder);
  }
}
```

Error

appSettings.json not found

Как найти appSettings.json

```
public class TestWebApplicationFactory : WebApplicationFactory<TestStartup> {
  protected override IWebHostBuilder CreateWebHostBuilder() {
    return WebHost.CreateDefaultBuilder().UseStartup<Startup>();
  }
  protected override void ConfigureWebHost(IWebHostBuilder builder){
    var config = new Dictionary<string, string>{
                    ["Domain:CopyrightYear"] = "42"
                };
    builder.ConfigureAppConfiguration(
                b => b.AddInMemoryCollection(config));
    base.ConfigureWebHost(builder);
  }
}
```

Как найти appSettings.json

```
public class TestWebApplicationFactory : WebApplicationFactory<TestStartup> {
  protected override IWebHostBuilder CreateWebHostBuilder() {
    return WebHost.CreateDefaultBuilder().UseStartup<Startup>();
  }
  protected override void ConfigureWebHost(IWebHostBuilder builder){

    var config = new Dictionary<string, string>{
                    ["Domain:CopyrightYear"] = "42"
                };
    builder.ConfigureAppConfiguration(
                b => b.AddInMemoryCollection(config));
    base.ConfigureWebHost(builder);
  }
}
```

Как найти appSettings.json

```
public class TestWebApplicationFactory : WebApplicationFactory<TestStartup> {
  protected override IWebHostBuilder CreateWebHostBuilder() {
    return WebHost.CreateDefaultBuilder().UseStartup<Startup>();
  }
  protected override void ConfigureWebHost(IWebHostBuilder builder){
    builder.UseContentRoot("../../../../MutantsCatalogue.Application");
    var config = new Dictionary<string, string>{
                    ["Domain:CopyrightYear"] = "42"
                };
    builder.ConfigureAppConfiguration(
                b => b.AddInMemoryCollection(config));
    base.ConfigureWebHost(builder);
  }
}
```

Как найти appSettings.json

```
public class TestWebApplicationFactory : WebApplicationFactory<TestStartup> {
  protected override IWebHostBuilder CreateWebHostBuilder() {
    return WebHost.CreateDefaultBuilder().UseStartup<Startup>();
  }
  protected override void ConfigureWebHost(IWebHostBuilder builder){
    builder.UseContentRoot("../../../../MutantsCatalogue.Application");
    var config = new Dictionary<string, string>{
                    ["Domain:CopyrightYear"] = "42"
                };
    builder.ConfigureAppConfiguration(
                b => b.AddInMemoryCollection(config));
    base.ConfigureWebHost(builder);
  }
}
```

Заглушки

DB

HTTP SERVICES

ETC.

DB: Entity Framework Core

Sqlite:memory

- Реляционная БД
- Изоляция в рамках connection
- Легко применим только при внедрении DbContext через конструктор

In-Memory

- Набор данных в памяти
- Изоляция в рамках процесса
- Можно использовать фабрики, virtual methods, etc.

Test#3: Get mutant

Заполнить In-Memory DB тестовыми данными

Передать In-Memory DbContext в приложение

Выполнить тест и проверить результат

Test#3.1: Fill In-memory DbContext

```
[Fact]
public async Task GetMutant_0003_Wolverine(){
  // arrange
  var options = new DbContextOptionsBuilder<MutantsContext>()
    .UseInMemoryDatabase(databaseName: nameof(GetMutant_0003_Wolverine)).Options;
  using (var context = new MutantsContext(options)){
    context.Mutants.Add(new Mutant{
        Name = "Wolverine",
        RealName = "Logan",
        Superpower = "Invulnerability, Claws"
    });
    context.SaveChanges();
  }
…
```

Test#3.1: Fill In-memory DbContext

```
[Fact]
public async Task GetMutant_0003_Wolverine(){
  // arrange
  var options = new DbContextOptionsBuilder<MutantsContext>()
    .UseInMemoryDatabase(databaseName: nameof(GetMutant_0003_Wolverine)).Options;
  using (var context = new MutantsContext(options)){
    context.Mutants.Add(new Mutant{
        Name = "Wolverine",
        RealName = "Logan",
        Superpower = "Invulnerability, Claws"
    });
    context.SaveChanges();
  }
```

…

Test#3.1: Fill In-memory DbContext

```
[Fact]
public async Task GetMutant_0003_Wolverine(){
  // arrange
  var options = new DbContextOptionsBuilder<MutantsContext>()
    .UseInMemoryDatabase(databaseName: nameof(GetMutant_0003_Wolverine)).Options;
  using (var context = new MutantsContext(options)){
    context.Mutants.Add(new Mutant{
        Name = "Wolverine",
        RealName = "Logan",
        Superpower = "Invulnerability, Claws"
    });
    context.SaveChanges();
  }
…
```

Test#3.2: Inject in-memory DbContext to SUT

- Переопределить abstract factory
  - Только если у вас используется данный паттерн
- If(test) условие
  - Можно использовать Configuration
  - Анти-паттерн
- Зарегистрировать в контейнере
  - WebApplicationFactory<T>.WithWebHostBuilder(…)
  - IWebHostBuilder. ConfigureTestServices(…)

Test#3.2: Inject in-memory DbContext to SUT

```
[Fact]
public async Task GetMutant_0003_Wolverine(){
  // arrange
  var options = new DbContextOptionsBuilder<MutantsContext>()
    .UseInMemoryDatabase(databaseName: nameof(GetMutant_0003_Wolverine)).Options;
  using (var context = new MutantsContext(options)){
    context.Mutants.Add(new Mutant{
        Name = "Wolverine",
        RealName = "Logan",
        Superpower = "Invulnerability, Claws"
    });
    context.SaveChanges();
  }
…
```

Test#3.2: Inject in-memory DbContext to SUT

```
[Fact]
public async Task GetMutant_0003_Wolverine(){
  // arrange
  var options = new DbContextOptionsBuilder<MutantsContext>()
    .UseInMemoryDatabase(databaseName: nameof(GetMutant_0003_Wolverine)).Options;
  using (var context = new MutantsContext(options)){
    …
  }
…
```

Test#3.2: Inject in-memory DbContext to SUT

```
[Fact]
public async Task GetMutant_0003_Wolverine(){
  // arrange
  var options = new DbContextOptionsBuilder<MutantsContext>()
    .UseInMemoryDatabase(databaseName: nameof(GetMutant_0003_Wolverine)).Options;
  using (var context = new MutantsContext(options)){
    …
  }
  var currentfactory = factory
        .WithWebHostBuilder(builder => builder.ConfigureTestServices(
            services => services.AddSingleton(new MutantsContext(options))));
  var client = currentfactory.CreateClient();
…
```

Test#3.2: Inject in-memory DbContext to SUT

```
[Fact]
public async Task GetMutant_0003_Wolverine(){
  // arrange
  var options = new DbContextOptionsBuilder<MutantsContext>()
    .UseInMemoryDatabase(databaseName: nameof(GetMutant_0003_Wolverine)).Options;
  using (var context = new MutantsContext(options)){
    …
  }
  var currentfactory = factory
        .WithWebHostBuilder(builder => builder.ConfigureTestServices(
            services => services.AddSingleton(new MutantsContext(options))));
  var client = currentfactory.CreateClient();
…
```

Test#3.2: Inject in-memory DbContext to SUT

```
[Fact]
public async Task GetMutant_0003_Wolverine(){
  // arrange
  var options = new DbContextOptionsBuilder<MutantsContext>()
    .UseInMemoryDatabase(databaseName: nameof(GetMutant_0003_Wolverine)).Options;
  using (var context = new MutantsContext(options)){
    …
  }
  var currentfactory = factory
        .WithWebHostBuilder(builder => builder.ConfigureTestServices(
            services => services.AddSingleton(new MutantsContext(options))));
  var client = currentfactory.CreateClient();
…
```

Test#3.2: Inject in-memory DbContext to SUT

```
[Fact]
public async Task GetMutant_0003_Wolverine(){
  // arrange
  var options = new DbContextOptionsBuilder<MutantsContext>()
    .UseInMemoryDatabase(databaseName: nameof(GetMutant_0003_Wolverine)).Options;
  using (var context = new MutantsContext(options)){
    …
  }
  var currentfactory = factory
        .WithWebHostBuilder(builder => builder.ConfigureTestServices(
            services => services.AddSingleton(new MutantsContext(options))));
  var client = currentfactory.CreateClient();
…
```

Test#3.2: Inject in-memory DbContext to SUT

```
[Fact]
public async Task GetMutant_0003_Wolverine(){
  // arrange
  var options = new DbContextOptionsBuilder<MutantsContext>()
    .UseInMemoryDatabase(databaseName: nameof(GetMutant_0003_Wolverine)).Options;
  using (var context = new MutantsContext(options)){
    …
  }
  var currentfactory = factory
        .WithWebHostBuilder(builder => builder.ConfigureTestServices(
            services => services.AddSingleton(new MutantsContext(options))));
  var client = currentfactory.CreateClient();
…
```

Test#3.3: Act and Assert

```
[Fact]
public async Task GetMutant_0003_Wolverine(){
  …
  // act
  var response = await client.GetAsync("api/mutants/Wolverine");
  // assert
  response.EnsureSuccessStatusCode();
  var result = await response.Content.ReadAsAsync<Mutant>();
  Assert.Equal("Wolverine", result.Name);
  Assert.Equal("Logan", result.RealName);
  Assert.Equal("Invulnerability, Claws", result.Superpower);
}
```

Epic combat mode

Request URL

```
http://localhost:58381/api/Combat/epic?attacker=Magneto&defender=Xavier
```

Server response

Code	Details
200	**Response body**

```
{
  "victoryPhrase": "Think in terms of opportunities and solutions instead of problems,
disappointment, and failure.",
}
```

QuotesProxy

```
public async Task<string> GetQuoteAsync(string category){
  string url = "https://quotes.rest/qod";
  if (category != null) url = $"{url}?category={category}";
  var request = new HttpRequestMessage(HttpMethod.Get, url);
  request.Headers.Add("Accept", "application/json");
  HttpClient client = clientFactory.CreateClient();
  var response = await client.SendAsync(request);
  if (response.IsSuccessStatusCode){
    var result = await response.Content.ReadAsAsync<Result>();
    return result?.Contents?.Quotes?.FirstOrDefault()?.Quote;
  }
  return null;
}
```

QuotesProxy

```
public async Task<string> GetQuoteAsync(string category){
  string url = "https://quotes.rest/qod";
  if (category != null) url = $"{url}?category={category}";
  var request = new HttpRequestMessage(HttpMethod.Get, url);
  request.Headers.Add("Accept", "application/json");
  HttpClient client = clientFactory.CreateClient();
  var response = await client.SendAsync(request);
  if (response.IsSuccessStatusCode){
    var result = await response.Content.ReadAsAsync<Result>();
    return result?.Contents?.Quotes?.FirstOrDefault()?.Quote;
  }
  return null;
}
```

QuotesProxy

```
public async Task<string> GetQuoteAsync(string category){
  string url = "https://quotes.rest/qod";
  if (category != null) url = $"{url}?category={category}";
  var request = new HttpRequestMessage(HttpMethod.Get, url);
  request.Headers.Add("Accept", "application/json");
  HttpClient client = clientFactory.CreateClient();
  var response = await client.SendAsync(request);
  if (response.IsSuccessStatusCode){
    var result = await response.Content.ReadAsAsync<Result>();
    return result?.Contents?.Quotes?.FirstOrDefault()?.Quote;
  }
  return null;
}
```

QuotesProxy: IHttpClientFactory

```
public async Task<string> GetQuoteAsync(string category){
  string url = "https://quotes.rest/qod";
  if (category != null) url = $"{url}?category={category}";
  var request = new HttpRequestMessage(HttpMethod.Get, url);
  request.Headers.Add("Accept", "application/json");
  HttpClient client = clientFactory.CreateClient();
  var response = await client.SendAsync(request);
  if (response.IsSuccessStatusCode){
    var result = await response.Content.ReadAsAsync<Result>();
    return result?.Contents?.Quotes?.FirstOrDefault()?.Quote;
  }
  return null;
}
```

QuotesProxy: IHttpClientFactory

```
public async Task<string> GetQuoteAsync(string category){
  string url = "https://quotes.rest/qod";
  if (category != null) url = $"{url}?category={category}";
  var request = new HttpRequestMessage(HttpMethod.Get, url);
  request.Headers.Add("Accept", "application/json");
  HttpClient client = clientFactory.CreateClient();
  var response = await client.SendAsync(request);
  if (response.IsSuccessStatusCode){
    var result = await response.Content.ReadAsAsync<Result>();
    return result?.Contents?.Quotes?.FirstOrDefault()?.Quote;
  }
  return null;
}
```

Startup

```
public void ConfigureServices(IServiceCollection services){
  services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version_2_2);
  services.AddHttpClient();
  ConfigureUtilityServices(services);
  …
  services.AddDomain();
  services.AddDal();
  services.AddInfrastructure();
}
```

Startup

```
public void ConfigureServices(IServiceCollection services){
  services.AddMvc().SetCompatibilityVersion(CompatibilityVersion.Version_2_2);
  services.AddHttpClient();
  ConfigureUtilityServices(services);
  …
  services.AddDomain();
  services.AddDal();
  services.AddInfrastructure();
}
```

Test#4: Victory Phrase

Mock HttpClient

Подложить в приложение

Выполнить тест

HttpServices: RichardSzalay.MockHttp

```
var mockHttp = new MockHttpMessageHandler();

// Setup a respond for the user api (including a wildcard in the URL)
mockHttp.When("http://localhost/api/user/*")
        .Respond("application/json", "{'name' : 'Test McGee'}"); // Respond with JSON

// Inject the handler or client into your application code
var client = mockHttp.ToHttpClient();

var response = await client.GetAsync("http://localhost/api/user/1234");
// or without async: var response = client.GetAsync("http://localhost/api/user/1234").Result;

var json = await response.Content.ReadAsStringAsync();

// No network connection required
Console.Write(json); // {'name' : 'Test McGee'}
```

Test#4.1: Mock Http

```
[Fact]
public async Task CombatEpic_0004_AnyMutant_ReturnsVictoryPhrase(){
  // arrange
  var mockHttp = new MockHttpMessageHandler();
  mockHttp.When(HttpMethod.Get, "https://quotes.rest/qod")
        .Respond("application/json", @"{
  'contents': {
    'quotes': [{
        'quote': 'The answer is 42'
      }
    ]
  }
}");
…
```

Test#4.1: Mock Http

```
[Fact]
public async Task CombatEpic_0004_AnyMutant_ReturnsVictoryPhrase(){
  // arrange
  var mockHttp = new MockHttpMessageHandler();
  mockHttp.When(HttpMethod.Get, "https://quotes.rest/qod")
        .Respond("application/json", @"{
  'contents': {
    'quotes': [{
        'quote': 'The answer is 42'
      }
    ]
  }
}");
…
```

Test#4.1: Mock Http

```
[Fact]
public async Task CombatEpic_0004_AnyMutant_ReturnsVictoryPhrase(){
  // arrange
  var mockHttp = new MockHttpMessageHandler();
  mockHttp.When(HttpMethod.Get, "https://quotes.rest/qod")
        .Respond("application/json", @"{
  'contents': {
    'quotes': [{
        'quote': 'The answer is 42'
      }
    ]
  }
}");
…
```

Test#4.1: Mock Http

```
[Fact]
public async Task CombatEpic_0004_AnyMutant_ReturnsVictoryPhrase(){
  // arrange
  var mockHttp = new MockHttpMessageHandler();
  mockHttp.When(HttpMethod.Get, "https://quotes.rest/qod")
        .Respond("application/json", @"{
  'contents': {
    'quotes': [{
        'quote': 'The answer is 42'
      }
    ]
  }
}");
```

…

Test#4.2: Inject to the application

```
[Fact]
public async Task CombatEpic_0004_AnyMutant_ReturnsVictoryPhrase(){
…
  var httpClientfactory = Substitute.For<IHttpClientFactory>();
  httpClientfactory.CreateClient(Arg.Any<string>())
        .Returns(mockHttp.ToHttpClient());
  var currentfactory = factory
        .WithWebHostBuilder(builder => builder.ConfigureTestServices(
            services => services.AddSingleton(httpClientfactory)));
…
```

Test#4.2: Inject to the application

```
[Fact]
public async Task CombatEpic_0004_AnyMutant_ReturnsVictoryPhrase(){
…
  var httpClientfactory = Substitute.For<IHttpClientFactory>();
  httpClientfactory.CreateClient(Arg.Any<string>())
        .Returns(mockHttp.ToHttpClient());
 var currentfactory = factory
        .WithWebHostBuilder(builder => builder.ConfigureTestServices(
            services => services.AddSingleton(httpClientfactory)));
…
```

Test#4.2: Inject to the application

```
[Fact]
public async Task CombatEpic_0004_AnyMutant_ReturnsVictoryPhrase(){
…
  var httpClientfactory = Substitute.For<IHttpClientFactory>();
  httpClientfactory.CreateClient(Arg.Any<string>())
        .Returns(mockHttp.ToHttpClient());
 var currentfactory = factory
        .WithWebHostBuilder(builder => builder.ConfigureTestServices(
            services => services.AddSingleton(httpClientfactory)));…
```

Test#4.2: Inject to the application

```
[Fact]
public async Task CombatEpic_0004_AnyMutant_ReturnsVictoryPhrase(){
…
  var httpClientfactory = Substitute.For<IHttpClientFactory>();
  httpClientfactory.CreateClient(Arg.Any<string>())
        .Returns(mockHttp.ToHttpClient());
 var currentfactory = factory
        .WithWebHostBuilder(builder => builder.ConfigureTestServices(
            services => services.AddSingleton(httpClientfactory)));…
```

Test#4.2: Inject to the application

```
[Fact]
public async Task CombatEpic_0004_AnyMutant_ReturnsVictoryPhrase(){
…
  var httpClientfactory = Substitute.For<IHttpClientFactory>();
  httpClientfactory.CreateClient(Arg.Any<string>())
        .Returns(mockHttp.ToHttpClient());
 var currentfactory = factory
        .WithWebHostBuilder(builder => builder.ConfigureTestServices(
            services => services.AddSingleton(httpClientfactory)));…
```

Test#4.3: Act and Assert

```
[Fact]
public async Task CombatEpic_0004_AnyMutant_ReturnsVictoryPhrase(){
  …
  var client = currentfactory.CreateClient();
  // act
  var response = await client
        .GetAsync("api/combat/epic?attacker=Magneto&defender=Xavier");
  // assert
  response.EnsureSuccessStatusCode();
  var result = await response.Content.ReadAsAsync<CombatResult>();
  Assert.Equal("The answer is 42", result.VictoryPhrase);
}
```

Test#4.3: Act and Assert

```
[Fact]
public async Task CombatEpic_0004_AnyMutant_ReturnsVictoryPhrase(){
  …
  var client = factory.CreateClient();
  // act
  var response = await client
        .GetAsync("api/combat/epic?attacker=Magneto&defender=Xavier");
  // assert
  response.EnsureSuccessStatusCode();
  var result = await response.Content.ReadAsAsync<CombatResult>();
  Assert.Equal("The answer is 42", result.VictoryPhrase);
}
```

Test#5: Wolverine Victory Phrase

Test#5: Wolverine Victory Phrase

```
[Fact]
public async Task CombatEpic_0005_Wolverine_ReturnsVictoryPhraseAboutLive() {
  // arrange
  var mockHttp = new MockHttpMessageHandler();
  mockHttp.Expect(HttpMethod.Get, "https://quotes.rest/qod")
    .WithQueryString("category", "life")
    .Respond("application/json",
                 @"{'contents':{'quotes':[{'quote':'Life is life'}]}}");
…
```

Test#5: Wolverine Victory Phrase

```
[Fact]
public async Task CombatEpic_0005_Wolverine_ReturnsVictoryPhraseAboutLive() {
  // arrange
  var mockHttp = new MockHttpMessageHandler();
  mockHttp.Expect(HttpMethod.Get, "https://quotes.rest/qod")
    .WithQueryString("category", "life")
    .Respond("application/json",
                 @"{'contents':{'quotes':[{'quote':'Life is life'}]}}");
…
```

Test#5: Wolverine Victory Phrase

```
[Fact]
public async Task CombatEpic_0005_Wolverine_ReturnsVictoryPhraseAboutLive() {
  // arrange
  var mockHttp = new MockHttpMessageHandler();
  mockHttp.Expect(HttpMethod.Get, "https://quotes.rest/qod")
    .WithQueryString("category", "life")
    .Respond("application/json",
                 @"{'contents':{'quotes':[{'quote':'Life is life'}]}}");
…
```

Test#5: Wolverine Victory Phrase

```
[Fact]
public async Task CombatEpic_0005_Wolverine_ReturnsVictoryPhraseAboutLive() {
  // arrange
  var mockHttp = new MockHttpMessageHandler();
  mockHttp.Expect(HttpMethod.Get, "https://quotes.rest/qod")
    .WithQueryString("category", "life")
    .Respond("application/json",
                 @"{'contents':{'quotes':[{'quote':'Life is life'}]}}");
…
```

Test#5: Wolverine Victory Phrase

```
…
  var httpClientfactory = Substitute.For<IHttpClientFactory>();
  httpClientfactory.CreateClient(Arg.Any<string>())
    .Returns(mockHttp.ToHttpClient());

  var currentfactory = factory
    .WithWebHostBuilder(builder => builder.ConfigureTestServices(
                    services => services.AddSingleton(httpClientfactory)));

  var client = currentfactory.CreateClient();
…
```

Test#5: Wolverine Victory Phrase

```
…
  var httpClientfactory = Substitute.For<IHttpClientFactory>();
  httpClientfactory.CreateClient(Arg.Any<string>())
    .Returns(mockHttp.ToHttpClient());

  var currentfactory = factory
    .WithWebHostBuilder(builder => builder.ConfigureTestServices(
                    services => services.AddSingleton(httpClientfactory)));

  var client = currentfactory.CreateClient();
…
```

Test#5: Wolverine Victory Phrase

```
…
  var httpClientfactory = Substitute.For<IHttpClientFactory>();
  httpClientfactory.CreateClient(Arg.Any<string>())
    .Returns(mockHttp.ToHttpClient());

  var currentfactory = factory
    .WithWebHostBuilder(builder => builder.ConfigureTestServices(
                    services => services.AddSingleton(httpClientfactory)));

  var client = currentfactory.CreateClient();
…
```

Test#5: Wolverine Victory Phrase

```
…
  var httpClientfactory = Substitute.For<IHttpClientFactory>();
  httpClientfactory.CreateClient(Arg.Any<string>())
    .Returns(mockHttp.ToHttpClient());

  var currentfactory = factory
    .WithWebHostBuilder(builder => builder.ConfigureTestServices(
                    services => services.AddSingleton(httpClientfactory)));

  var client = currentfactory.CreateClient();
…
```

Test#5: Wolverine Victory Phrase

```
…
  // act
  var response = await client
    .GetAsync("api/combat/epic?attacker=Wolverine&defender=Beast");

  // assert
  mockHttp.VerifyNoOutstandingExpectation();
}
```

Test#5: Wolverine Victory Phrase

```
…
  // act
  var response = await client
    .GetAsync("api/combat/epic?attacker=Wolverine&defender=Beast");

  // assert
  mockHttp.VerifyNoOutstandingExpectation();
}
```

Test#5: Wolverine Victory Phrase

```
[Fact]
public async Task CombatEpic_0005_Wolverine_ReturnsVictoryPhraseAboutLive() {
  // arrange
  var mockHttp = new MockHttpMessageHandler();
  mockHttp.Expect(HttpMethod.Get, "https://quotes.rest/qod")
    .WithQueryString("category", "life")
    .Respond("application/json",
                 @"{'contents':{'quotes':[{'quote':'Life is life'}]}}");
…
  // assert
  mockHttp.VerifyNoOutstandingExpectation();
}
```

Test#5: Wolverine Victory Phrase

```
[Fact]
public async Task CombatEpic_0005_Wolverine_ReturnsVictoryPhraseAboutLive() {
  // arrange
  var mockHttp = new MockHttpMessageHandler();
  mockHttp.Expect(HttpMethod.Get, "https://quotes.rest/qod")
    .WithQueryString("category", "life")
    .Respond("application/json",
                 @"{'contents':{'quotes':[{'quote':'Life is life'}]}}");
…
  // assert
  mockHttp.VerifyNoOutstandingExpectation();
}
```

Test#5: Wolverine Victory Phrase

```
[Fact]
public async Task CombatEpic_0005_Wolverine_ReturnsVictoryPhraseAboutLive() {
  // arrange
  var mockHttp = new MockHttpMessageHandler();
  mockHttp.Expect(HttpMethod.Get, "https://quotes.rest/qod")
    .WithQueryString("category", "life")
    .Respond("application/json",
                 @"{'contents':{'quotes':[{'quote':'Life is life'}]}}");
…
  // assert
  mockHttp.VerifyNoOutstandingExpectation();
}
```

MockHttp и autorest

```
public ApiClient(HttpClient httpClient,
    bool disposeHttpClient)
    : base(httpClient, disposeHttpClient)
{
  Initialize();
}
```

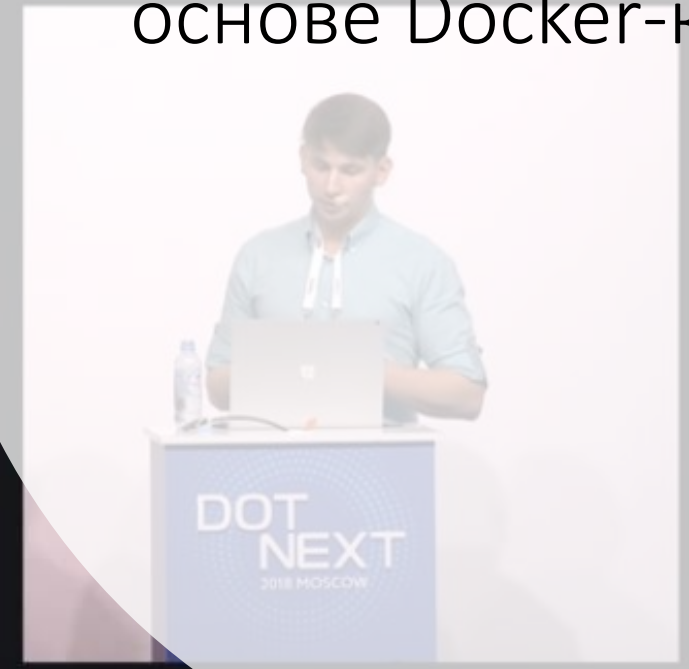
Другие зависимости

- Использовать заглушки
- Положиться на локальную инфраструктуру
- Docker

DOTNEXT 2018

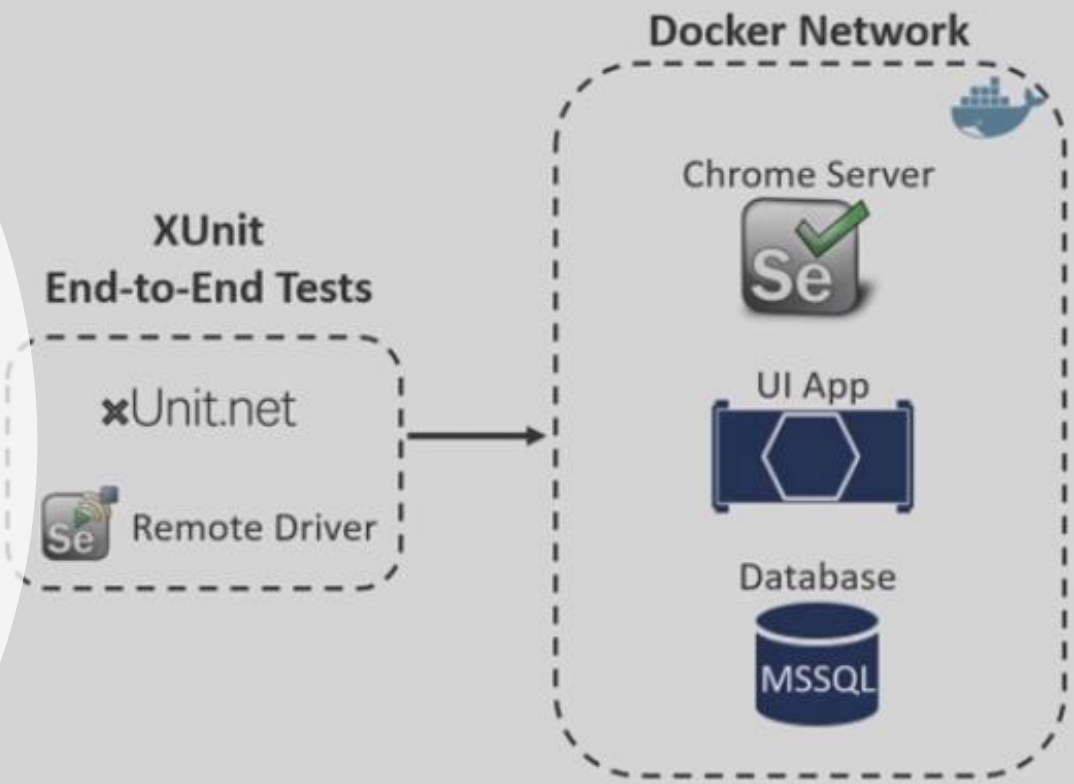
22-23 Н

Алексей Горшколеп — Создание окружения для интеграционных тестов на основе Docker-контейнеров



12
2

End-to-End-Testing with Docker



Ubiquitous Language + Gherkin Syntax

- Терминология бизнес домена

+

- Scenario
- Steps
  - Given
  - When
  - Then

Результаты тестов можно показывать не техническим людям

Gherkin Syntax

- Scenario – Fact/Test/TestMethod
- Steps
  - Given – Arrange
  - When – Act
  - Then – Assert

#1: Wolverine vs Magneto

```
[Fact]
public async Task Combat_MagnetoVsWolverine_MagnetoWins()
{
  // arrange
  var client = factory.CreateClient();
  // act
  var response = await client.
              GetAsync("api/combat?attacker=Magneto&defender=Wolverine");
  // assert
  response.EnsureSuccessStatusCode();
  var result = await response.Content.ReadAsAsync<CombatResult>();
  Assert.Equal("Magneto", result.Winner);
}
```

#1: Wolverine vs Magneto

- When combat between Magneto and Wolverine
- Then winner is Magneto

#1: Wolverine vs Magneto

```
[Scenario]
public void Combat_0001_MagnetoVsWolverine_MagnetoWins(HttpResponseMessage response){
  $"When combat between Magneto and Wolverine«
      .x(async () => response = await factory.CreateClient()
            .GetAsync("api/combat?attacker=Magneto&defender=Wolverine"));
  "Then winner is Magneto".x(async () =>
    {
      response.EnsureSuccessStatusCode();
      var result = await response.Content.ReadAsAsync<CombatResult>();
      Assert.Equal("Magneto", result.Winner);
    });
}
```

#1: Wolverine vs Magneto

```
[Scenario]
public void Combat_0001_MagnetoVsWolverine_MagnetoWins(HttpResponseMessage response){
  $"When combat between Magneto and Wolverine«
      .x(async () => response = await factory.CreateClient()
            .GetAsync("api/combat?attacker=Magneto&defender=Wolverine"));
  "Then winner is Magneto".x(async () =>
    {
      response.EnsureSuccessStatusCode();
      var result = await response.Content.ReadAsAsync<CombatResult>();
      Assert.Equal("Magneto", result.Winner);
    });
}
```

#1: Wolverine vs Magneto

```
[Scenario]
public void Combat_0001_MagnetoVsWolverine_MagnetoWins(HttpResponseMessage response){
  $"When combat between Magneto and Wolverine«
      .x(async () => response = await factory.CreateClient()
            .GetAsync("api/combat?attacker=Magneto&defender=Wolverine"));
  "Then winner is Magneto".x(async () =>
    {
      response.EnsureSuccessStatusCode();
      var result = await response.Content.ReadAsAsync<CombatResult>();
      Assert.Equal("Magneto", result.Winner);
    });
}
```

#1: Wolverine vs Magneto

```
[Scenario]
public void Combat_0001_MagnetoVsWolverine_MagnetoWins(HttpResponseMessage response){
  $"When combat between Magneto and Wolverine«
      .x(async () => response = await factory.CreateClient()
            .GetAsync("api/combat?attacker=Magneto&defender=Wolverine"));
  "Then winner is Magneto".x(async () =>
    {
      response.EnsureSuccessStatusCode();
      var result = await response.Content.ReadAsAsync<CombatResult>();
      Assert.Equal("Magneto", result.Winner);
    });
}
```

#1: Wolverine vs Magneto

```
[Scenario]
public void Combat_0001_MagnetoVsWolverine_MagnetoWins(HttpResponseMessage response){
  $"When combat between Magneto and Wolverine«
      .x(async () => response = await factory.CreateClient()
            .GetAsync("api/combat?attacker=Magneto&defender=Wolverine"));
  "Then winner is Magneto".x(async () =>
    {
      response.EnsureSuccessStatusCode();
      var result = await response.Content.ReadAsAsync<CombatResult>();
      Assert.Equal("Magneto", result.Winner);
    });
}
```

#1: Wolverine vs Magneto

```
[Scenario]
public void Combat_0001_MagnetoVsWolverine_MagnetoWins(HttpResponseMessage response){
  $"When combat between Magneto and Wolverine«
      .x(async () => response = await factory.CreateClient()
            .GetAsync("api/combat?attacker=Magneto&defender=Wolverine"));
  "Then winner is Magneto".x(async () =>
    {
      response.EnsureSuccessStatusCode();
      var result = await response.Content.ReadAsAsync<CombatResult>();
      Assert.Equal("Magneto", result.Winner);
    });
}
```

#1: Wolverine vs Magneto

```
[Scenario]
public void Combat_0001_MagnetoVsWolverine_MagnetoWins(HttpResponseMessage response){
  $"When combat between Magneto and Wolverine«
      .x(async () => response = await factory.CreateClient()
            .GetAsync("api/combat?attacker=Magneto&defender=Wolverine"));
  "Then winner is Magneto".x(async () =>
    {
      response.EnsureSuccessStatusCode();
      var result = await response.Content.ReadAsAsync<CombatResult>();
      Assert.Equal("Magneto", result.Winner);
    });
}
```

#1: Wolverine vs Magneto

```
[Scenario]
public void Combat_0001_MagnetoVsWolverine_MagnetoWins(HttpResponseMessage response){
  $"When combat between Magneto and Wolverine«
      .x(async () => response = await factory.CreateClient()
            .GetAsync("api/combat?attacker=Magneto&defender=Wolverine"));
  "Then winner is Magneto".x(async () =>
    {
      response.EnsureSuccessStatusCode();
      var result = await response.Content.ReadAsAsync<CombatResult>();
      Assert.Equal("Magneto", result.Winner);
    });
}
```

Test#3: Get mutant

```
[Fact]
public async Task GetMutant_0003_Wolverine(){
  // arrange
  var options = new DbContextOptionsBuilder<MutantsContext>()
      .UseInMemoryDatabase(nameof(GetMutant_0003_Wolverine))
      .Options;
  using (var context = new MutantsContext(options)){
    context.Mutants.Add(new Mutant{
          Name = "Wolverine",
          RealName = "Logan",
          Superpower = "Invulnerability, Claws"
    });
    context.SaveChanges();
  }
```

```
  var currentfactory = factory
      .WithWebHostBuilder(builder =>
          builder.ConfigureTestServices(
              services => services.AddSingleton(
                          new MutantsContext(options))));

  var client = currentfactory.CreateClient();
  // act
  var response = await client.GetAsync("api/mutants/Wolverine");
  // assert
  response.EnsureSuccessStatusCode();
  result = await response.Content.ReadAsAsync<Mutant>();
  Assert.Equal("Wolverine", result.Name);
  Assert.Equal("Logan", result.RealName);
  Assert.Equal("Invulnerability, Claws", result.Superpower);
}
```

Test#3: Get mutant

- Given database
- And the mutant Wolverine (Logan): Invulnerability, Claws
- When get mutant by name Wolverine
- Then returns the mutant Wolverine (Logan): Invulnerability, Claws

Test#3.1: Given database

```
[Scenario]
public void GetMutant_0003_Wolverine(DbContextOptions<MutantsContext> options,
  WebApplicationFactory<TestStartup> currentfactory, HttpResponseMessage response){

  $"Given database".x(() => {
      options = new DbContextOptionsBuilder<MutantsContext>()
           .UseInMemoryDatabase(databaseName: nameof(GetMutant_0003_Wolverine))
           .Options;
      });
  …
```

Test#3.1: Given database

```
[Scenario]
public void GetMutant_0003_Wolverine(DbContextOptions<MutantsContext> options,
  WebApplicationFactory<TestStartup> currentfactory, HttpResponseMessage response){

  $"Given database".x(() => {
      options = new DbContextOptionsBuilder<MutantsContext>()
           .UseInMemoryDatabase(databaseName: nameof(GetMutant_0003_Wolverine))
           .Options;
      });
  …
```

Test#3.1: Given database

```
[Scenario]
public void GetMutant_0003_Wolverine(DbContextOptions<MutantsContext> options,
  WebApplicationFactory<TestStartup> currentfactory, HttpResponseMessage response){

  $"Given database".x(() => {
      options = new DbContextOptionsBuilder<MutantsContext>()
           .UseInMemoryDatabase(databaseName: nameof(GetMutant_0003_Wolverine))
           .Options;
      });
  …
```

Test#3.2: And the mutant …

```
var mutant = new Mutant {
    Name = "Wolverine",
    RealName = "Logan",
    Superpower = "Invulnerability, Claws"
};
$"And the mutant {mutant}".x(() => {
    using (var context = new MutantsContext(options)) {
      context.Mutants.Add(mutant);
      context.SaveChanges();
    }
    currentfactory = factory
         .WithWebHostBuilder(builder => builder.ConfigureTestServices(
                services => services.AddSingleton(new MutantsContext(options))));
    });
```

…

Test#3.2: And the mutant …

```
var mutant = new Mutant {
    Name = "Wolverine",
    RealName = "Logan",
    Superpower = "Invulnerability, Claws"
};
$"And the mutant {mutant}".x(() => {
    using (var context = new MutantsContext(options)) {
      context.Mutants.Add(mutant);
      context.SaveChanges();
    }
    currentfactory = factory
         .WithWebHostBuilder(builder => builder.ConfigureTestServices(
                services => services.AddSingleton(new MutantsContext(options))));
    });
…
```

Test#3.2: And the mutant …

```
var mutant = new Mutant {
    Name = "Wolverine",
    RealName = "Logan",
    Superpower = "Invulnerability, Claws"
};
$"And the mutant {mutant}".x(() => {
    using (var context = new MutantsContext(options)) {
      context.Mutants.Add(mutant);
      context.SaveChanges();
    }
    currentfactory = factory
         .WithWebHostBuilder(builder => builder.ConfigureTestServices(
                services => services.AddSingleton(new MutantsContext(options))));
    });
…
```

Test#3.2: And the mutant …

```
var mutant = new Mutant {
    Name = "Wolverine",
    RealName = "Logan",
    Superpower = "Invulnerability, Claws"
};
$"And the mutant {mutant}".x(() => {
    using (var context = new MutantsContext(options)) {
      context.Mutants.Add(mutant);
      context.SaveChanges();
    }
    currentfactory = factory
         .WithWebHostBuilder(builder => builder.ConfigureTestServices(
                services => services.AddSingleton(new MutantsContext(options))));
    });
```

…

Test#3.2: And the mutant …

```
var mutant = new Mutant {
    Name = "Wolverine",
    RealName = "Logan",
    Superpower = "Invulnerability, Claws"
};
$"And the mutant {mutant}".x(() => {
    using (var context = new MutantsContext(options)) {
      context.Mutants.Add(mutant);
      context.SaveChanges();
    }
    currentfactory = factory
         .WithWebHostBuilder(builder => builder.ConfigureTestServices(
                services => services.AddSingleton(new MutantsContext(options))));
    });
```

…

Test#3.2: And the mutant …

```
var mutant = new Mutant {
    Name = "Wolverine",
    RealName = "Logan",
    Superpower = "Invulnerability, Claws"
};
$"And the mutant {mutant}".x(() => {
    using (var context = new MutantsContext(options)) {
      context.Mutants.Add(mutant);
      context.SaveChanges();
    }
    currentfactory = factory
         .WithWebHostBuilder(builder => builder.ConfigureTestServices(
                services => services.AddSingleton(new MutantsContext(options))));
    });
```

…

Test#3.3: When …

```
…
 "When get mutant by name Wolverine"
        .x(async () => response = await currentfactory.CreateClient()
                .GetAsync("api/mutants/Wolverine"));
…
```

Test#3.3: When …

```
…
  "When get mutant by name Wolverine"
        .x(async () => response = await currentfactory.CreateClient()
                .GetAsync("api/mutants/Wolverine"));
…
```

Test#3.4: Then returns the mutant …

```
…
 "Then returns the mutant Wolverine (Logan): Invulnerability, Claws"
     .x(async () =>{
         response.EnsureSuccessStatusCode();
         var result = await response.Content.ReadAsAsync<Mutant>();
         Assert.Equal("Wolverine", result.Name);
         Assert.Equal("Logan", result.RealName);
         Assert.Equal("Invulnerability, Claws", result.Superpower);
     });
}
```

Test#3.4: Then returns the mutant …

```
…
 "Then returns the mutant Wolverine (Logan): Invulnerability, Claws"
     .x(async () =>{
         response.EnsureSuccessStatusCode();
         var result = await response.Content.ReadAsAsync<Mutant>();
         Assert.Equal("Wolverine", result.Name);
         Assert.Equal("Logan", result.RealName);
         Assert.Equal("Invulnerability, Claws", result.Superpower);
     });
}
```

Test#3.4: Then returns the mutant …

```
…
 "Then returns the mutant Wolverine (Logan): Invulnerability, Claws"
      .x(async () =>{
          response.EnsureSuccessStatusCode();
          var result = await response.Content.ReadAsAsync<Mutant>();
          Assert.Equal("Wolverine", result.Name);
          Assert.Equal("Logan", result.RealName);
          Assert.Equal("Invulnerability, Claws", result.Superpower);
      });
}
```

Функциональные тесты WebApi – это

- Быстро в написании
  - WebApplicationFactory<T>
- Вкусно в возможностях
  - Entity Framework Core In-Memory/Sqlite
  - MockHttp
  - …
- Не дорого, а очень дорого. Но
  - Очевидно для клиента
  - Удобно для QA и BA
  - Упрощает поддержку

Ссылки

- Integration tests in ASP.NET Core: https://docs.microsoft.com/en-us/aspnet/core/test/integration-tests?view=aspnetcore-2.2
- Ef Core Testing: https://docs.microsoft.com/en-us/ef/core/miscellaneous/testing/
- MockHttp: https://github.com/richardszalay/mockhttp
- Gherkin Syntax: https://docs.cucumber.io/gherkin/
- xBehave: http://xbehave.github.io/
- https://github.com/AleksandrKugushev/presentations/tree/master/2019_Functional-Testing

Спасибо!