

Razvoj primijenjene programske potpore

11. Testiranje

Općenito o testiranju

- *„Unit testing is a tool and not a religion and only you know how much testing you require”*
Adam Freeman: Pro ASP.NET Core MVC, Sixth Edition, Apress
- *„Testing can only show the presence of bugs, not their absence”*
Edsger W. Dijkstra
- Praktični aspekti testiranja
 - regresijsko testiranje – testiranjem se može brzo provjeriti da promjenama nije narušen prethodno ispravni kod koji je već bio testiran, tj da nije došlo do regresije
 - predstavlja jedan vid dokumentacije kojom se opisuje očekivano ponašanje sustava
 - automatizirati testiranje dijelova koje je nepraktično ili dugotrajno za ručno provjeriti
 - fokusirati se na komplicirane i važnije dijelove, a ne na trivijalnosti - pokrivenost koda testovima je mjera koja može zavarati

Vrste testiranja korištene u primjerima

- Jedinično testiranje
 - Testiraju se dijelovi koda, tj. razredi i njihova definirana ponašanja
 - Jedinični test
 - mora biti brz (mjereno u milisekundama)
 - mora se moći pokrenuti izolirano od vanjskih ovisnosti
 - mora se moći ponoviti i davati konzistentne rezultate
- Integracijsko testiranje
 - Testiraju se (prethodno jedinično testirani) elementi, odnosno ispravnost njihove međusobne integracije
- U primjerima se ilustrira upotreba ovih tehnika za MVC aplikaciju
 - Ostale vrste testiranja, tehnike odabira reprezentativnih testnih podataka itd... izlaze van okvira ovih predavanja i nisu predmet razmatranja, što ne znači da ih ne treba uzeti u obzir.

Radni okviri za testiranje u .NET okruženju

- *xUnit, NUnit, MSBuild*
 - <https://docs.microsoft.com/en-us/dotnet/core/testing/#testing-tools>
- Podržani kroz *TestExplorer* u *Visual Studiu*
- U primjerima koji slijedi koristi se *xUnit*, ali koncepti su isti
 - <https://xunit.net/docs/comparisons>
 - <https://www.lambdatest.com/blog/nunit-vs-xunit-vs-mstest/>
- Redoslijed u testu:
Arrange – Act – Assert

The screenshot shows the Test Explorer window in Visual Studio. The top bar indicates 26 tests passed, 24 succeeded, and 2 failed. The main table lists tests with their durations and traits. The 'Category [DocumentFilter] (8)' test is expanded, showing a failure in 'MVC.UnitTests.ViewModels.DocumentFilterTests.FilterString_Is_WellFormed'. The Test Detail Summary below shows the failure message: 'Assert.Equal() Failure' with expected and actual values for a date string.

Test	Duration	Traits	Error Message
Category [AutoCompleteController] .	2,3 sec		
Category [CitiesController] (1)	1,4 sec		
Category [CountriesController] (3)	1,5 sec		
Category [DocumentFilter] (8)	15 ms		
DocumentFilterTests (8)	15 ms		
MVC.UnitTests.ViewModels.Doc...	< 1 ms	Category [...]	
MVC.UnitTests.ViewModels.Doc...	6 ms	Category [...]	Assert.Equal() I
MVC.UnitTests.ViewModels.D...	6 ms	Category [...]	Assert.Equal() I
MVC.UnitTests.ViewModels.D...	< 1 ms	Category [...]	Assert.Equal() I
MVC.UnitTests.ViewModels.D...	< 1 ms	Category [...]	
MVC.UnitTests.ViewModels.Doc...	< 1 ms	Category [...]	
MVC.UnitTests.ViewModels.Doc...	9 ms	Category [...]	
MVC.UnitTests.ViewModels.Doc...	< 1 ms		
MVC.UnitTests.ViewModels.D...	< 1 ms	Category [...]	
MVC.UnitTests.ViewModels.D...	< 1 ms	Category [...]	
Category [Integration Tests] (7)	9,4 sec		


Test Detail Summary

- MVC.UnitTests.ViewModels.DocumentFilterTests.FilterString_Is_WellFormed
 - Source: [DocumentFilterTests.cs](#) line 51
 - Duration: < 1 ms
 - Message:
 - Assert.Equal() Failure
 - ↓ (pos 16)
 - Expected: 1-15.12.2015-07.5.2021-300-500
 - Actual: 1-15.12.2015-07.05.2021-300-500
 - ↑ (pos 16)
 - Stack Trace:
 - [DocumentFilterTests.FilterString_Is_WellFormed\(String expect](#)

Imenovanje testova


- Različite prakse i preporuke ovisno o izvoru
 - <https://dzone.com/articles/7-popular-unit-test-naming>
 - *MethodName_StateUnderTest_ExpectedBehavior*
 - *MethodName_Scenario_ExpectedBehavior*
<https://docs.microsoft.com/en-us/dotnet/core/testing/unit-testing-best-practices>
 - *MethodName_ExpectedBehavior_StateUnderTest*
 - *test[Feature being tested]*
 - *Feature to be tested*
 - *Should_ExpectedBehavior_When_StateUnderTest*
 - *When_StateUnderTest_Expect_ExpectedBehavior*
 - *Given_Preconditions_When_StateUnderTest_Then_ExpectedBehavior*
 - Organizacija i imenovanje razreda i prostora imena
 - <https://ardalis.com/unit-test-naming-convention/>

Primjer jediničnog testiranja

- Skupom testnih postupaka testira se ispravnost pretvorbe aktivnog filtra u string kod primjera s dokumentima i obrnuto
- Test je ispravan ako su sve tvrdnje zadovoljene
 - Različiti postupci iz razreda *Assert*: *IsTrue*, *Equal*, *Contains*, *IsType*, ...
 - Testni postupak označen atributom *Fact*, a atributom *Trait* može se dodatno opisati što može biti korisno kod grupiranja u prikazu svih testova
 - Primjer:  Testing \ MVC.UnitTests \ ViewModels \ DocumentFilterTests

```
[Trait("Category", "DocumentFilter")]  
[Fact]  
public void EmptyFilter_Returns_4_Slashes() {  
    string expected = "----";  
    DokumentFilter filter = new();  
    string actual = filter.ToString();  
    Assert.Equal(expected, actual);  
}
```

Provjera bacanja iznimke

- Ponekad očekivano ponašanje treba biti bacanje iznimke
- Umjesto pisanja try-catch blokova, koristi se `Assert.Throws` u kombinaciji s akcijom koja treba proizvesti iznimku određenog tipa
 - Primjer:  `Testing \ MVC.UnitTests \ ViewModels \ DocumentFilterTests`

[Fact]


```
public void InvalidDate_Throws_FormatException() {  
    string filterString = "-15.12.2015-15.13.2021--";  
    Assert.Throws<FormatException>(() =>  
        DokumentFilter.FromString(filterString));  
}
```

- Napomena: Test s *Throws* neće proći ako je generirana iznimka izvedena iz tražene, npr. sljedeća tvrdnja ne bi bila zadovoljena

```
Assert.Throws<Exception>(() =>  
    DokumentFilter.FromString(filterString));
```

- Za tu namjenu postoji *ThrowsAny*

Ponavljanje istog testa s različitim podacima

- Ponekad je isti test potrebno ponoviti s različitim argumentima, pa se umjesto s [Fact] test može označiti s [Theory], a u [InlineData] navesti vrijednosti koje će se pridružiti ulaznim argumentima testnog postupka
 - Primjer:  Testing \ MVC.UnitTests \ ViewModels \ DocumentFilterTests


```
[Theory]
[InlineData("1-15.12.2015-15.12.2021-300")]
[InlineData("1-2-3-4-5-6")]
public void Not_4_Slashes_IsEmptyFilter(string filterString) {
    DokumentFilter filter = DokumentFilter.FromString(filterString);
    Assert.True(filter.IsEmpty());
}
```

- Broj argumenata može biti proizvoljan, ali se u ***InlineData*** mogu navoditi **samo konstante**.
 - string, int, double, ... ali ne i decimal, float i slično (bitno ako je ulazni argument nulabilan) – vidi testni postupak *FilterString_Is_WellFormed* u istom razredu

Dodatni alati za pisanje tvrdnji

- Fluent Assertions <https://fluentassertions.com/>
 - NuGet paket *FluentAssertions*
 - Omogućava pisanje provjera koje izgledaju prirodnije ljudskom jeziku
 - `variable.Should().Be(value)`
 - `variable.Should().BeGreaterThan(value, because: "some reason...");`
 - `variable.Should().
 .NotBeNull()
 .And.BeOfType<List<int>>
 .Which.Count().Should().Be(value)`
 - Bit će korišteni u primjerima koji slijede


Kreiranje instance upravljača kod testiranja (1)

- Zahtijevane ovisnosti se (uobičajeno) navode u konstruktoru pojedinog upravljača
 - U skladu s principom *Explicit Dependencies Principle*
<https://deviq.com/principles/explicit-dependencies-principle>
 - ASP.NET Core automatski umetne potrebne ovisnosti kod instanciranja upravljača s ciljem izvršavanja akcije
 - U testovima upravljač instanciramo sami, pa moramo i sami pripremiti objekte o kojima upravljač ovisi
 - Primjer:  ...\MVC.UnitTests\Controllers\AutoComplete\Mjesto.cs

```
public async Task ReturnsCity_UsingCaseInsensitiveSubstring(
    string term, string expectedCity, string unexpectedCity) {


    using var ctx = new FirmaContext(dbContextBuilder.Options);
    var controller = new AutoCompleteController(ctx, options);
    IEnumerable<IdLabel> result = await controller.Mjesto(term)
    ...
}
```

Kreiranje instance upravljača kod testiranja (2)

- U konstruktoru podesimo postavke za instanciranje konteksta
 - Definiran konfiguracijski objekt s postavkama za spajanje na bazu podataka
 - Primjer:  ...UnitTests\Controllers\AutoComplete\Mjesto.cs


```
public class Mjesto{
    private DbContextOptionsBuilder<FirmaContext> dbContextBuilder;
    public Mjesto() {
        //Arrange
        var builder = new ConfigurationBuilder()
            ...
        var configuration = builder.Build();
        dbContextBuilder = new DbContextOptionsBuilder<FirmaContext>()
            .UseSqlServer(configuration.GetConnectionString("Firma"));
        ...
    }
}
```

Kreiranje instance upravljača kod testiranja (3)

- Što ako trebamo samo dio objekta o kojem ovisimo? Npr. u aplikaciji koju testiramo uspostavili smo preslikavanje između dijela konfiguracijske datoteke
 - U dijelu koji testiramo, koristimo samo informaciju o max. broju rezultata
 - Možemo koristiti imitacije objekata (*Mock*) i definirati vlastito ponašanje samo pojedinih postupaka
 - NuGet paket *Moq*
 - Primjer:  ...UnitTests\Controllers\AutoComplete\Mjesto.cs


```
public class Mjesto{  
    private readonly IOptionsSnapshot<AppSettings> options;  
    public Mjesto() { ...  
        var mockOptions = new Mock<IOptionsSnapshot<AppSettings>>();  
        var appSettings = new AppSettings {  
            AutoCompleteCount = 10  
        };  
        mockOptions.SetupGet(options => options.Value)  
            .Returns(appSettings);  
        options = mockOptions.Object  
    }
```

Testna baza podataka u memoriji

- Upravljač *Drzava* pri izvođenju akcije *Index* treba preusmjeriti korisnika na akciju *Create* ako u bazi podataka nema podataka
 - Praznu bazu podataka ćemo osigurati time što ćemo stvoriti EF bazu podataka u memoriji
 - Više testova može dijeliti istu bazu podataka u memoriji, pa jedinstvenost osiguramo imenom i postavljamo u testnom postupku, a ne u konstruktoru
 - Primjer:  Testing \ MVC.UnitTests \ ViewModels \ DocumentFilterTests


```
public void RedirectsToCreate_WhenNoCountries() {  
    ...  
    var dbOptions = new DbContextOptionsBuilder<FirmaContext>()  
        .UseInMemoryDatabase(  
            databaseName: nameof(RedirectsToCreate_WhenNoCountries))  
        .Options;  
    using var context = new FirmaContext(dbOptions);  
    ...  
}
```

Testiranje upravljača i *TempData*

- Upravljač *Drzava* pri izvođenju akcije *Index* treba preusmjeriti korisnika na akciju *Create* ako u bazi podataka nema podataka
 - Upravljač zapisuje informaciju u *TempData* koji je posljedica postojanja sjednice na serveru.
 - U jediničnom testiranju *TempData* treba zamijeniti proizvoljnom implementacijom ili imitirati
 - Primjer:  ...UnitTests\Controllers\Drzava\Index


```
public Index() {  
    var tempDataMock = new Mock<ITempDataDictionary>();  
    tempData = tempDataMock.Object;  
    ...  
  
public void RedirectsToCreate_WhenNoCountries() {  
    ...  
    var controller = new DrzavaController(context, ...);  
    controller.TempData = tempData;  
}
```

Provjera tipa rezultata akcije upravljača

- Upravljač *Drzava* pri izvođenju akcije *Index* treba preusmjeriti korisnika na akciju *Create* ako u bazi podataka nema podataka
 - Provjeravamo je li rezultat akcije očekivan
 - Primjer:  ...UnitTests\Controllers\Drzava\Index



```
var controller = new DrzavaController(context, options,  
                                     mockLogger.Object);  
controller.TempData = tempData;  
  
var result = controller.Index();  
  
var redirectToActionResult =  
    Assert.IsType<RedirectToActionResult>(result);  
Assert.Equal("Create", redirectToActionResult.ActionName);
```

Potvrda interakcije s imitiranim objektima

- Upravljač *Drzava* pri izvođenju akcije *Index* treba preusmjeriti korisnika na akciju *Create* ako u bazi podataka nema podataka i zapisati o tome informaciju koristeći umetnuti *ILogger*
 - Provjeramo je li prilikom izvršavanja akcija pozvana odgovarajuća metoda i to s očekivanim parametrima (konkretno *LogInformation*)
 - *LogInformation* je pokrata za poziv *Log* i razinom poruke *Information*
 - Primjer:  ...UnitTests\Controllers\Drzava\Index

```
public void RedirectsToCreate_WhenNoCountries() {  
    ...  
    var result = controller.Index();  
    mockLogger.Verify(1 => 1.Log(LogLevel.Information,  
        It.IsAny<EventId>(),  
        It.IsAny<object>(),  
        It.IsAny<Exception>(),  
        (Func<object, Exception, string>)It.IsAny<object>()),  
        Times.Once());  
}
```



Na upravljaču se može testirati npr. i...

- ... je li upravljač vratio ispravan model i točne podatke
 - Podatke možemo pripremiti s memorijskom bazom podataka ili nekom drugom testnom bazom podataka
 - Primjer:  ...UnitTests\Controllers\Drzava\Index - Returns_CorrectPageData
- ... je li upravljač pripremio podatke za padajuću listu tako što ih je stavio u ViewBag/ViewData
 - Primjer:  ...UnitTests\Controllers\Mjesto\Create - PreparesViewBag

Integracijsko testiranje


- U prethodnim primjerima testirani su upravljači, a ovisnosti su bile eksplicitno definirane, a često i imitirane.
- Integracijskim testiranjem se provjerava rade li upravljači kad se integriraju s ostalim komponentama u web-server
 - Odazivaju li se na pravoj adresi?
 - Primaju li ispravno postavke za spajanje na bazu podataka?
 - Vraćaju li podatke u ispravnom formatu?
- U testu se simulira web-server korištenjem NuGet paketa *Microsoft.AspNetCore.Mvc.Testing*
 - Web server nastaje korištenjem objekta tipa *WebApplicationFactory<TEntryPoint>*
 - koristi se konfiguracijski razred iz testiranog projekta, pa je to *WebApplicationFactory<Program>*
 - potrebno dodati *public partial class Program {}* u Program.cs

xUnit i IClassFixture

- „Implementiranjem” sučelja *IClassFixture* (sučelje bez metoda) osigurava se zajednički objekt za sve testne postupke u tom razredu te je on umetnut u konstruktor.
 - U primjeru je to primjerak `WebApplicationFactory<Program>` koji omogućava stvaranje `HttpClienta` za pozive prema testnom serveru
 - Primjer:  ...IntegrationTests\AutoCompleteShould.cs


```
public class HomeControllerShould :  
    IClassFixture<WebApplicationFactory<Program>> {  
    private readonly WebApplicationFactory<Program> factory;  
  
    public HomeControllerShould(  
        WebApplicationFactory<Program> factory) {  
        this.factory = factory;  
    }  
    ... //kasnije HttpClient = factory.CreateClient();
```

Početna provjera integriranosti komponenti

- Odaziva li se aplikacija na početnoj stranici i vraća li HTML u kojem se nalazi riječ RPPP?
 - Primjer:  ...IntegrationTests\AutoCompleteShould.cs
 - Primijetiti da se ne poziva upravljač kroz kod, već otvara HTTP veza prema određenoj adresi i ispituje odgovor

```
[Theory]
[InlineData("Home/Index")]
[InlineData("")]
public async Task ServeHomePage(string url) {
    var client = factory.CreateClient();
    var response = await client.GetAsync(url);
    response.StatusCode.Should().Be(HttpStatusCode.OK);
    string content = await response.Content.ReadAsStringAsync();
    content.Should().Contain("RPPP");
}
```


Provjera serijalizacije

- Integracijsko testiranje bi npr. moglo otkriti da json koji nastane iz popisa gradova ima npr. neispravan *casing*, pa potencijalno deserijalizacija ne bi bila ispravna.
 - Primjer:  ...IntegrationTests\AutoCompleteShould.cs

```
[Theory] [InlineData("velika gor", 10410, 10380)]
public async Task ReturnCities(string input, params int[] postcodes) {
    string url = "/AutoComplete/Mjesto?term=" + input;
    var client = factory.CreateClient();
    var response = await client.GetAsync(url);
    response.StatusCode.Should().Be(HttpStatusCode.OK);
    var stream = await response.Content.ReadAsStreamAsync();
    var data = await
        JsonSerializer.DeserializeAsync<IEnumerable<IdLabel>>(stream);
    ... //imamo li barem 2 zapisa u data i to one koji sadrže
    // navedene nazive i poštanske brojeve
```


- Ukloniti *JsonPropertyName* iz *IdLabel*, pa pokrenuti ponovo test

Korištenje memorijske baze podataka u integracijskom testiranju (1)

- Integracijsko testiranje koristi stvarne komponentne koje su postavljene u aplikaciji, ali moguće je neke ukloniti/promijeniti nadjačavanjem postupaka iz *WebApplicationFactory*
 - Primjer:  ...IntegrationTests\CustomWebApplicationFactory.cs

```
public class CustomWebApplicationFactory<TTest> :  
    WebApplicationFactory<Program> {  
    ... override void ConfigureWebHost(IWebHostBuilder builder) {  
        builder.ConfigureServices(services => {  
            var descriptor = services.SingleOrDefault(  
d => d.ServiceType == typeof(DbContextOptions<FirmaContext>));  
            services.Remove(descriptor);  
  
            services.AddDbContext<FirmaContext>(options => {  
                options.UseInMemoryDatabase("Firma-" +  
                    typeof(TTest).Name); //database per test collection  
            });  
        });  
    }  
}
```

Korištenje memorijske baze podataka u integracijskom testiranju (2) ...

- ... nam omogućava da lako pripremimo situaciju kojom možemo provjeriti ispravnost pojedinih scenarija, npr. preusmjerenja s Index na Create
- Primjer:  ...IntegrationTests\CountryControllerShould.cs

```
public class CountryControllerShould :  
    IClassFixture<CustomWebApplicationFactory<CountryControllerShould>>{  
    ...  
    [Fact]  
    public async Task Redirect_When_DbContainsNoCountries() {  
        string url = "/Drzava/Index";  
        var client = factory.CreateClient(new  
            WebApplicationFactoryClientOptions {  
                AllowAutoRedirect = false  
            });  
        var response = await client.GetAsync(url);  
        response.StatusCode.Should().Be(HttpStatusCode.Redirect);  
        response.Headers.Location.Should().NotNull()  
            .And.Be("/Drzava/Create");  
    }  
}
```

Napomene uz EF i memorijske baze podataka

- Treba imati na umu da se memorijska baza podataka za Entity Framework ne mora ponašati kao ostale relacijske baze podataka
 - case sensitive
 - ne podržava transakcije
 - ne podržava direktne SQL upite

<https://docs.microsoft.com/en-us/ef/core/testing/#approach-3-the-ef-core-in-memory-database>
- Poželjno izbjegavati za integracijsko testiranje, odnosno zamijeniti s SQLite
 - <https://jimmybogard.com/avoid-in-memory-databases-for-tests/>
 - <https://docs.microsoft.com/en-us/ef/core/testing/in-memory>