Getting Started
Startup Templates
Tutorials
Web Application Development
1: Creating the Server Side
2: The Book List Page
3: Creating, Updating and Deleting Books
4: Integration Tests
5: Authorization
6: Authors: Domain layer
7: Authors: Database Integration
8: Authors: Application Layer
9: Authors: User Interface
10: Book to Author Relation
Community Articles
Migrating from the ASP.NET Boilerplate
Fundamentals
Infrastructure
Architecture
API
User Interface
Data Access
Real Time
Testing
Samples
Application Modules
Release Information
Reference
Contribution Guide

In this document

This document has multiple versions. Select the options best fit for you.

UI MVC / Ra Database Er

Web Application Development Tutorial - Part 4: Integration Tests

About This Tutorial

In this tutorial series, you will build an ABP based web application named `Acme.BookStore`. This application is used to manage a list of books and their authors. It is developed using the following technologies:

- **Entity Framework Core** as the ORM provider.
- **MVC / Razor Pages** as the UI Framework.

This tutorial is organized as the following parts;

- Part 1: Creating the server side
- Part 2: The book list page
- Part 3: Creating, updating and deleting books
- **Part 4: Integration tests (this part)**
- Part 5: Authorization
- Part 6: Authors: Domain layer
- Part 7: Authors: Database Integration
- Part 8: Authors: Application Layer
- Part 9: Authors: User Interface
- Part 10: Book to Author Relation

Download the Source Code

This tutorial has multiple versions based on your **UI** and **Database** preferences. We've prepared a few combinations of the source code to be downloaded:

- MVC (Razor Pages) UI with EF Core
- Blazor UI with EF Core
- Angular UI with MongoDB

Video Tutorial

This part is also recorded as a video tutorial and **published on YouTube**.

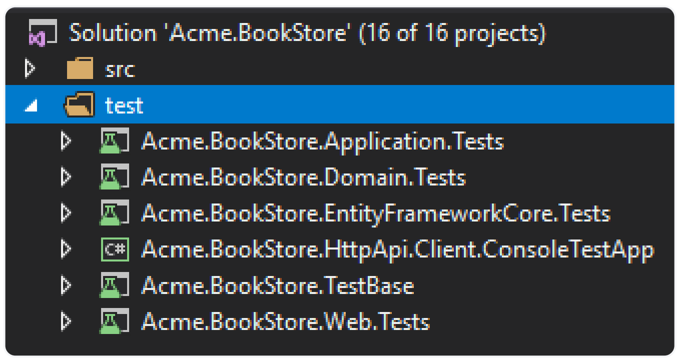
Test Projects in the Solution

This part covers the **server side** tests. There are several test projects in the solution:

4.1 (latest) English

Filter topics

- Getting Started
- Startup Templates
- Tutorials
  - Web Application Development
    - 1: Creating the Server Side
    - 2: The Book List Page
    - 3: Creating, Updating and Deleting Books
    - 4: Integration Tests
    - 5: Authorization
    - 6: Authors: Domain layer
    - 7: Authors: Database Integration
    - 8: Authors: Application Layer
    - 9: Authors: User Interface
    - 10: Book to Author Relation
  - Community Articles
  - Migrating from the ASP.NET Boilerplate
- Fundamentals
- Infrastructure
- Architecture
- API
- User Interface
- Data Access
- Real Time
- Testing
- Samples
- Application Modules
- Release Information
- Reference
- Contribution Guide


```
Solution 'Acme.BookStore' (16 of 16 projects)
  src
  test
    Acme.BookStore.Application.Tests
    Acme.BookStore.Domain.Tests
    Acme.BookStore.EntityFrameworkCore.Tests
    Acme.BookStore.HttpApi.Client.ConsoleTestApp
    Acme.BookStore.TestBase
    Acme.BookStore.Web.Tests
```

Share on :

In this document

> Test projects slightly differs based on your UI and Database selection. For example, if you select MongoDB, then the `Acme.BookStore.EntityFrameworkCore.Tests` will be `Acme.BookStore.MongoDB.Tests` .

Each project is used to test the related project. Test projects use the following libraries for testing:

- Xunit as the main test framework.
- Shoudly as the assertion library.
- NSubstitute as the mocking library.

> The test projects are configured to use **SQLite in-memory** as the database. A separate database instance is created and seeded (with the data seed system) to prepare a fresh database for every test.

Adding Test Data

If you had created a data seed contributor as described in the first part, the same data will be available in your tests. So, you can skip this section. If you haven't created the seed contributor, you can use the `BookStoreTestDataSeedContributor` to seed the same data to be used in the tests below.

Testing the BookAppService

Add a new test class, named `BookAppService_Tests` in the `Books` namespace (folder) of the `Acme.BookStore.Application.Tests` project:

4.1 (latest) English

Filter topics

- Getting Started
- Startup Templates
- Tutorials
  - Web Application Development
    - 1: Creating the Server Side
    - 2: The Book List Page
    - 3: Creating, Updating and Deleting Books
    - 4: Integration Tests
    - 5: Authorization
    - 6: Authors: Domain layer
    - 7: Authors: Database Integration
    - 8: Authors: Application Layer
    - 9: Authors: User Interface
    - 10: Book to Author Relation
  - Community Articles
  - Migrating from the ASP.NET Boilerplate
- Fundamentals
- Infrastructure
- Architecture
- API
- User Interface
- Data Access
- Real Time
- Testing
- Samples
- Application Modules
- Release Information
- Reference
- Contribution Guide

```
using System;
using System.Linq;
using System.Threading.Tasks;
using Shouldly;
using Volo.Abp.Application.Dtos;
using Volo.Abp.Validation;
using Xunit;

namespace Acme.BookStore.Books
{
    public class BookAppService_Tests : BookStoreApplic
    {
        private readonly IBookAppService _bookAppServic

        public BookAppService_Tests()
        {
            _bookAppService = GetRequiredService<IBookA
        }

        [Fact]
        public async Task Should_Get_List_Of_Books()
        {
            //Act
            var result = await _bookAppService.GetListA
                new PagedAndSortedResultRequestDto()
            );

            //Assert
            result.TotalCount.ShouldBeGreaterThan(0);
            result.Items.ShouldContain(b => b.Name == "
        }
    }
}
```

- `Should_Get_List_Of_Books` test simply uses `BookAppService.GetListAsync` method to get and check the list of books.
- We can safely check the book "1984" by its name, because we know that this books is available in the database since we've added it in the seed data.

Add a new test method to the `BookAppService_Tests` class that creates a new **valid** book:

Share on :

In this document

4.1 (latest) English

Filter topics

- Getting Started
- Startup Templates
- Tutorials
  - Web Application Development
    - 1: Creating the Server Side
    - 2: The Book List Page
    - 3: Creating, Updating and Deleting Books
    - 4: Integration Tests
    - 5: Authorization
    - 6: Authors: Domain layer
    - 7: Authors: Database Integration
    - 8: Authors: Application Layer
    - 9: Authors: User Interface
    - 10: Book to Author Relation
  - Community Articles
  - Migrating from the ASP.NET Boilerplate
- Fundamentals
- Infrastructure
- Architecture
- API
- User Interface
- Data Access
- Real Time
- Testing
- Samples
- Application Modules
- Release Information
- Reference
- Contribution Guide

```
[Fact]
public async Task Should_Create_A_Valid_Book()
{
    //Act
    var result = await _bookAppService.CreateAsync(
        new CreateUpdateBookDto
        {
            Name = "New test book 42",
            Price = 10,
            PublishDate = DateTime.Now,
            Type = BookType.ScienceFiction
        }
    );

    //Assert
    result.Id.ShouldNotBe(Guid.Empty);
    result.Name.ShouldBe("New test book 42");
}
```

Add a new test that tries to create an invalid book and fails:

```
[Fact]
public async Task Should_Not_Create_A_Book_Without_Name
{
    var exception = await Assert.ThrowsAsync<AbpValidat
    {
        await _bookAppService.CreateAsync(
            new CreateUpdateBookDto
            {
                Name = "",
                Price = 10,
                PublishDate = DateTime.Now,
                Type = BookType.ScienceFiction
            }
        );
    });

    exception.ValidationErrors
        .ShouldContain(err => err.MemberNames.Any(mem =
}
```

- Since the `Name` is empty, ABP will throw an `AbpValidationException`.

The final test class should be as shown below:

Share on :

In this document

4.1 (latest) English

Filter topics

- Getting Started
- Startup Templates
- Tutorials
  - Web Application Development
    - 1: Creating the Server Side
    - 2: The Book List Page
    - 3: Creating, Updating and Deleting Books
    - 4: Integration Tests
    - 5: Authorization
    - 6: Authors: Domain layer
    - 7: Authors: Database Integration
    - 8: Authors: Application Layer
    - 9: Authors: User Interface
    - 10: Book to Author Relation
  - Community Articles
  - Migrating from the ASP.NET Boilerplate
- Fundamentals
- Infrastructure
- Architecture
- API
- User Interface
- Data Access
- Real Time
- Testing
- Samples
- Application Modules
- Release Information
- Reference
- Contribution Guide

```
using System;
using System.Linq;
using System.Threading.Tasks;
using Shouldly;
using Volo.Abp.Application.Dtos;
using Volo.Abp.Validation;
using Xunit;

namespace Acme.BookStore.Books
{
    public class BookAppService_Tests : BookStoreApplic
    {
        private readonly IBookAppService _bookAppServic

        public BookAppService_Tests()
        {
            _bookAppService = GetRequiredService<IBookA
        }

        [Fact]
        public async Task Should_Get_List_Of_Books()
        {
            //Act
            var result = await _bookAppService.GetListA
                new PagedAndSortedResultRequestDto()
            );

            //Assert
            result.TotalCount.ShouldBeGreaterThan(0);
            result.Items.ShouldContain(b => b.Name == "
        }

        [Fact]
        public async Task Should_Create_A_Valid_Book()
        {
            //Act
            var result = await _bookAppService.CreateAs
                new CreateUpdateBookDto
                {
                    Name = "New test book 42",
                    Price = 10,
                    PublishDate = DateTime.Now,
                    Type = BookType.ScienceFiction
                }
            );

            //Assert
            result.Id.ShouldNotBe(Guid.Empty);
            result.Name.ShouldBe("New test book 42");
        }

        [Fact]
        public async Task Should_Not_Create_A_Book_With
        {
            var exception = await Assert.ThrowsAsync<Ab
            {
                await _bookAppService.CreateAsync(
                    new CreateUpdateBookDto
                    {
                        Name = "",
                        Price = 10,
```

Share on :

In this document

4.1 (latest) English

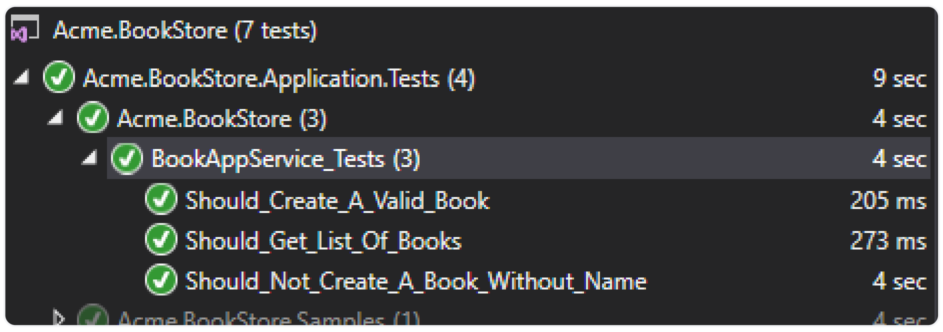
Filter topics

Getting Started
Startup Templates
Tutorials
Web Application Development
1: Creating the Server Side
2: The Book List Page
3: Creating, Updating and Deleting Books
4: Integration Tests
5: Authorization
6: Authors: Domain layer
7: Authors: Database Integration
8: Authors: Application Layer
9: Authors: User Interface
10: Book to Author Relation
Community Articles
Migrating from the ASP.NET Boilerplate
Fundamentals
Infrastructure
Architecture
API
User Interface
Data Access
Real Time
Testing
Samples
Application Modules
Release Information
Reference
Contribution Guide

```
                    PublishDate = DateTime.Now,
                    Type = BookType.ScienceFiction
                }
            );
        });

        exception.ValidationErrors
            .ShouldContain(err => err.MemberNames.A
    }
  }
}
```

Open the **Test Explorer Window** (use Test -> Windows -> Test Explorer menu if it is not visible) and **Run All** tests:



```
Acme.BookStore (7 tests)
◢ ✓ Acme.BookStore.Application.Tests (4)          9 sec
  ◢ ✓ Acme.BookStore (3)                          4 sec
    ◢ ✓ BookAppService_Tests (3)                  4 sec
        ✓ Should_Create_A_Valid_Book              205 ms
        ✓ Should_Get_List_Of_Books                273 ms
        ✓ Should_Not_Create_A_Book_Without_Name   4 sec
  ▷ ✓ Acme.BookStore.Samples (1)                  4 sec
```

Congratulations, the **green icons** indicates that the tests have been successfully passed!

The Next Part

See the next part of this tutorial.

Share on :

In this document