

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](#)

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

UI

MVC / Razor Pages

Database

Entity Framework Core

Web Application Development Tutorial - Part 5: Authorization

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](#)
- **Part 5: Authorization (this part)**
- [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](#).

Permissions

ABP Framework provides an [authorization system](#) based on the ASP.NET Core's [authorization infrastructure](#). One major feature added on top of the standard authorization infrastructure is the **permission system** which allows to define permissions and enable/disable per role, user or client.

Permission Names

In this document

 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](#)

A permission must have a unique name (a `string`). The best way is to define it as a `const`, so we can reuse the permission name.

Open the `BookStorePermissions` class inside the `Acme.BookStore.Application.Contracts` project (in the `Permissions` folder) and change the content as shown below:

```
namespace Acme.BookStore.Permissions
{
    public static class BookStorePermissions
    {
        public const string GroupName = "BookStore";

        public static class Books
        {
            public const string Default = GroupName + ".Books";
            public const string Create = Default + ".Create";
            public const string Edit = Default + ".Edit";
            public const string Delete = Default + ".Delete";
        }
    }
}
```

This is a hierarchical way of defining permission names. For example, "create book" permission name was defined as `BookStore.Books.Create`. ABP doesn't force you to a structure, but we find this way useful.

Permission Definitions

You should define permissions before using them.

Open the `BookStorePermissionDefinitionProvider` class inside the `Acme.BookStore.Application.Contracts` project (in the `Permissions` folder) and change the content as shown below:

Share on : [Twitter](#) [LinkedIn](#) [Email](#)

In this document

 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](#)

Share on : [Twitter](#) [LinkedIn](#) [Email](#)

In this document

```
using Acme.BookStore.Localization;
using Volo.Abp.Authorization.Permissions;
using Volo.Abp.Localization;

namespace Acme.BookStore.Permissions
{
    public class BookStorePermissionDefinitionProvider
    {
        public override void Define(IPermissionDefinitionContext context)
        {
            var bookStoreGroup = context.AddGroup(BookStorePermissions.Default);

            var booksPermission = bookStoreGroup.AddPermission(BookStorePermissions.Books);
            booksPermission.AddChild(BookStorePermissions.Books.Create);
            booksPermission.AddChild(BookStorePermissions.Books.Edit);
            booksPermission.AddChild(BookStorePermissions.Books.Delete);
        }

        private static LocalizableString L(string name)
        {
            return LocalizableString.Create<BookStoreResource>(name);
        }
    }
}
```

This class defines a **permission group** (to group permissions on the UI, will be seen below) and **4 permissions** inside this group. Also, **Create**, **Edit** and **Delete** are children of the `BookStorePermissions.Books.Default` permission. A child permission can be selected **only if the parent was selected**.

Finally, edit the localization file (`en.json` under the `Localization/BookStore` folder of the `Acme.BookStore.Domain.Shared` project) to define the localization keys used above:

```
"Permission:BookStore": "Book Store",
"Permission:Books": "Book Management",
"Permission:Books.Create": "Creating new books",
"Permission:Books.Edit": "Editing the books",
"Permission:Books.Delete": "Deleting the books"
```

Localization key names are arbitrary and no forcing rule. But we prefer the convention used above.

Permission Management UI

Once you define the permissions, you can see them on the **permission management modal**.

Go to the *Administration -> Identity -> Roles* page, select *Permissions* action for the admin role to open the permission management modal:

> [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](#)

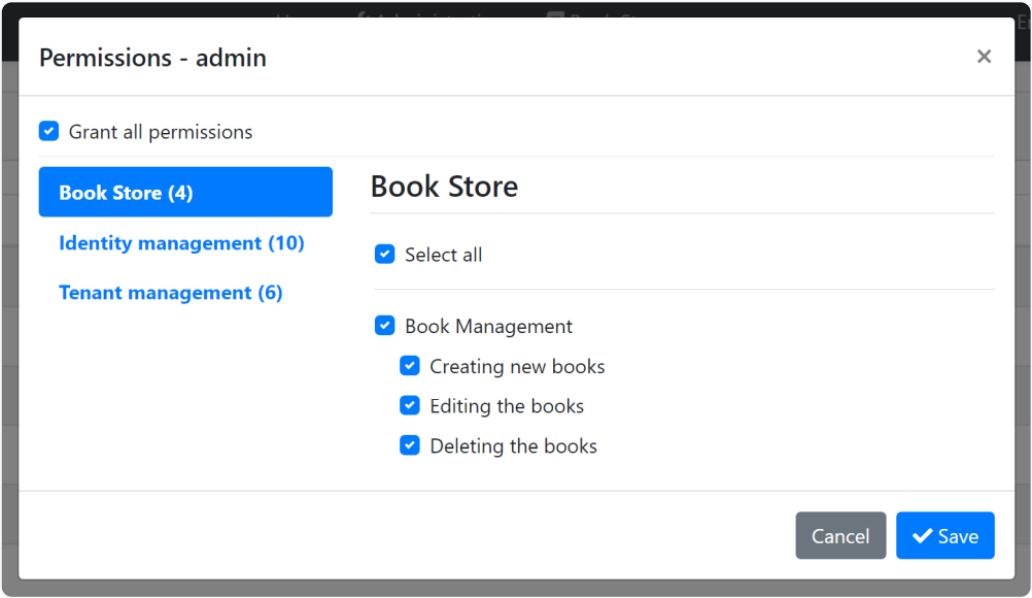
> [Application Modules](#)

> [Release Information](#)

> [Reference](#)

→ [Contribution Guide](#)

In this document



Grant the permissions you want and save the modal.

Tip: New permissions are automatically granted to the admin role if you run the `Acme.BookStore.DbMigrator` application.

Authorization

Now, you can use the permissions to authorize the book management.

Application Layer & HTTP API

Open the `BookAppService` class and add set the policy names as the permission names defined above:

```
using System;
using Acme.BookStore.Permissions;
using Volo.Abp.Application.Dtos;
using Volo.Abp.Application.Services;
using Volo.Abp.Domain.Repositories;

namespace Acme.BookStore.Books
{
    public class BookAppService :
        CrudAppService<
            Book, //The Book entity
            BookDto, //Used to show books
            Guid, //Primary key of the book entity
            PagedAndSortedResultRequestDto, //Used for
            CreateUpdateBookDto>, //Used to create/upda
            IBookAppService //implement the IBookAppService
    {
        public BookAppService(IRepository<Book, Guid> repository)
            : base(repository)
        {
            GetPolicyName = BookStorePermissions.Books.
            GetListPolicyName = BookStorePermissions.Bo
            CreatePolicyName = BookStorePermissions.Boo
            UpdatePolicyName = BookStorePermissions.Boo
            DeletePolicyName = BookStorePermissions.Boo
        }
    }
}
```

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**

Added code to the constructor. Base `CrudAppService` automatically uses these permissions on the CRUD operations. This makes the **application service** secure, but also makes the **HTTP API** secure since this service is automatically used as an HTTP API as explained before (see [auto API controllers](#)).

You will see the declarative authorization, using the `[Authorize(...)]` attribute, later while developing the author management functionality.

Razor Page

While securing the HTTP API & the application service prevents unauthorized users to use the services, they can still navigate to the book management page. While they will get authorization exception when the page makes the first AJAX call to the server, we should also authorize the page for a better user experience and security.

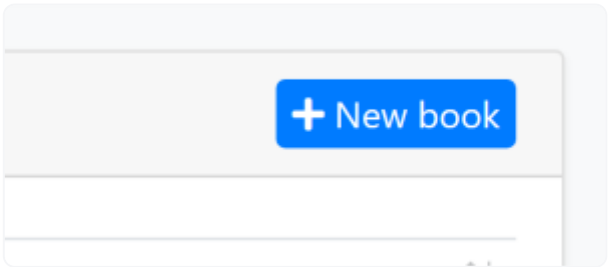
Open the `BookStoreWebModule` and add the following code block inside the `ConfigureServices` method:

```
Configure<RazorPagesOptions>(options =>
{
    options.Conventions.AuthorizePage("/Books/Index", B
    options.Conventions.AuthorizePage("/Books/CreateMod
    options.Conventions.AuthorizePage("/Books/EditModal
});
```

Now, unauthorized users are redirected to the **login page**.

Hide the New Book Button

The book management page has a *New Book* button that should be invisible if the current user has no *Book Creation* permission.



Open the `Pages/Books/Index.cshtml` file and change the content as shown below:

Share on : [Twitter](#) [LinkedIn](#) [Email](#)

In this document

 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](#)

Share on : [Twitter](#) [LinkedIn](#) [Email](#)

In this document

```
@page
@using Acme.BookStore.Localization
@using Acme.BookStore.Permissions
@using Acme.BookStore.Web.Pages.Books
@using Microsoft.AspNetCore.Authorization
@using Microsoft.Extensions.Localization
@model IndexModel
@Inject IStringLocalizer<BookStoreResource> L
@Inject IAuthorizationService AuthorizationService
@section scripts
{
    <abp-script src="/Pages/Books/Index.js"/>
}

<abp-card>
    <abp-card-header>
        <abp-row>
            <abp-column size-md="_6">
                <abp-card-title>@L["Books"]</abp-card-title>
            </abp-column>
            <abp-column size-md="_6" class="text-right">
                @if (await AuthorizationService.IsGrantedAsync(BookStorePermissions.Books.Create))
                {
                    <abp-button id="NewBookButton"
                                text="@L["NewBook"].Value"
                                icon="plus"
                                button-type="Primary"/>
                }
            </abp-column>
        </abp-row>
    </abp-card-header>
    <abp-card-body>
        <abp-table striped-rows="true" id="BooksTable">
        </abp-card-body>
    </abp-card>
```

- Added `@inject IAuthorizationService AuthorizationService` to access to the authorization service.
- Used `@if (await AuthorizationService.IsGrantedAsync(BookStorePermissions.Books.Create))` to check the book creation permission to conditionally render the *New Book* button.

JavaScript Side

Books table in the book management page has an actions button for each row. The actions button includes *Edit* and *Delete* actions:

> **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**

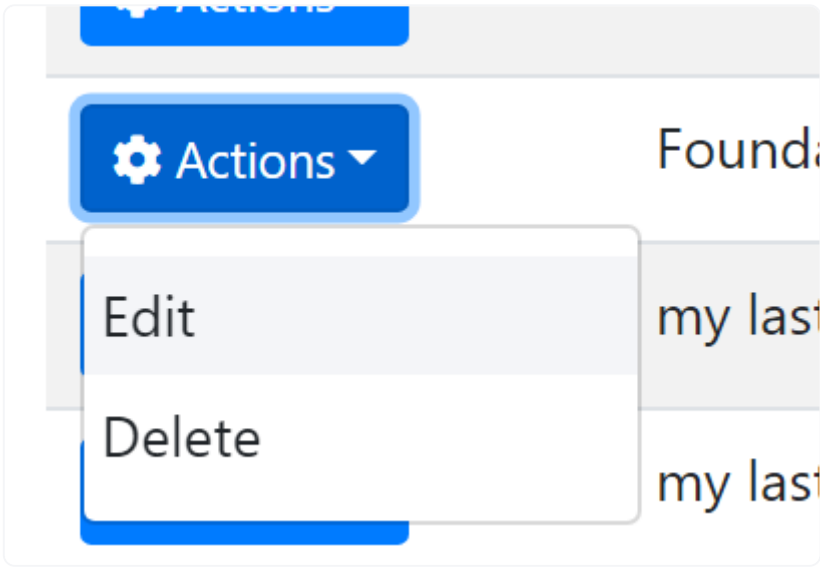
> **Reference**

> **Contribution Guide**

Share on :

In this document



We should hide an action if the current user has not granted for the related permission. Datatables row actions has a `visible` option that can be set to `false` to hide the action item.

Open the `Pages/Books/Index.js` inside the `Acme.BookStore.Web` project and add a `visible` option to the `Edit` action as shown below:

```
{
  text: l('Edit'),
  visible: abp.auth.isGranted('BookStore.Books.Edit')
  action: function (data) {
    editModal.open({ id: data.record.id });
  }
}
```

Do same for the `Delete` action:

```
visible: abp.auth.isGranted('BookStore.Books.Delete')
```

- `abp.auth.isGranted(...)` is used to check a permission that is defined before.
- `visible` could also be get a function that returns a `bool` if the value will be calculated later, based on some conditions.

Menu Item

Even we have secured all the layers of the book management page, it is still visible on the main menu of the application. We should hide the menu item if the current user has no permission.

Open the `BookStoreMenuContributor` class, find the code block below:

https://docs.abp.io/en/abp/latest/Tutorials/Part-5?UI=MVC&DB=EF

7/10

 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](#)

Share on : [Twitter](#) [LinkedIn](#) [Email](#)

In this document

```
context.Menu.AddItem(  
    new ApplicationMenuItem(  
        "BooksStore",  
        l["Menu:BookStore"],  
        icon: "fa fa-book"  
    ).AddItem(  
        new ApplicationMenuItem(  
            "BooksStore.Books",  
            l["Menu:Books"],  
            url: "/Books"  
        )  
    )  
);
```

And replace this code block with the following:

```
var bookStoreMenu = new ApplicationMenuItem(  
    "BooksStore",  
    l["Menu:BookStore"],  
    icon: "fa fa-book"  
);  
  
context.Menu.AddItem(bookStoreMenu);  
  
//CHECK the PERMISSION  
if (await context.IsGrantedAsync(BookStorePermissions.B  
{  
    bookStoreMenu.AddItem(new ApplicationMenuItem(  
        "BooksStore.Books",  
        l["Menu:Books"],  
        url: "/Books"  
    ));  
}
```

You also need to add `async` keyword to the `ConfigureMenuAsync` method and re-arrange the return values. The final `BookStoreMenuContributor` class should be the following:

 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](#)

Share on : [!\[\]\(dcadc17c064c775919616fcc152162e9_img.jpg\)](#) [!\[\]\(d0ea58d474979ad3390c9c5a943aca8c_img.jpg\)](#) [!\[\]\(1492590b7f30f49d1e3a7fe562ab91c0_img.jpg\)](#)

In this document

```
using System.Threading.Tasks;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Localization;
using Acme.BookStore.Localization;
using Acme.BookStore.MultiTenancy;
using Acme.BookStore.Permissions;
using Volo.Abp.TenantManagement.Web.Navigation;
using Volo.Abp.UI.Navigation;

namespace Acme.BookStore.Web.Menus
{
    public class BookStoreMenuContributor : IMenuContributor
    {
        public async Task ConfigureMenuAsync(MenuConfigurationContext context)
        {
            if (context.Menu.Name == StandardMenus.Main)
            {
                await ConfigureMainMenuAsync(context);
            }
        }

        private async Task ConfigureMainMenuAsync(MenuConfigurationContext context)
        {
            if (!MultiTenancyConsts.IsEnabled)
            {
                var administration = context.Menu.GetAdministration();
                administration.TryRemoveMenuItem(TenantManagement.Web.Navigation.Administration);
            }

            var l = context.GetLocalizer<BookStoreResource>();

            context.Menu.Items.Insert(0, new ApplicationMenuItem(
                "BooksStore",
                l["Menu:BookStore"],
                icon: "fa fa-book"
            ));

            context.Menu.AddItem(bookStoreMenu);

            //CHECK the PERMISSION
            if (await context.IsGrantedAsync(BookStorePermissions.ViewBooks))
            {
                bookStoreMenu.AddItem(new ApplicationMenuItem(
                    "BooksStore.Books",
                    l["Menu:Books"],
                    url: "/Books"
                ));
            }
        }
    }
}
```

The Next Part

See the [next part](#) of this tutorial.

 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**

In this document