

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 9: Authors: User Interface

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](#)
- [Part 6: Authors: Domain layer](#)
- [Part 7: Authors: Database Integration](#)
- [Part 8: Authors: Application Layer](#)
- **Part 9: Authors: User Interface (this part)**
- [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](#)

Introduction

This part explains how to create a CRUD page for the **Author** entity introduced in the previous parts.

The Authors List Page

Create a new razor page, **Index.cshtml** under the **Pages/Authors** folder of the **Acme.BookStore.Web** project and change the content as given below.

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**
- # Index.cshtml
- Share on :   
- ## In this document
- ```
@page
@using Acme.BookStore.Localization
@using Acme.BookStore.Permissions
@using Acme.BookStore.Web.Pages.Authors
@using Microsoft.AspNetCore.Authorization
@using Microsoft.Extensions.Localization
@inject IStringLocalizer<BookStoreResource> L
@inject IAuthorizationService AuthorizationService
@model IndexModel

@section scripts
{
 <abp-script src="/Pages/Authors/Index.js"/>
}

<abp-card>
 <abp-card-header>
 <abp-row>
 <abp-column size-md="_6">
 <abp-card-title>@L["Authors"]</abp-card-title>
 </abp-column>
 <abp-column size-md="_6" class="text-right">
 @if (await AuthorizationService
 .IsGrantedAsync(BookStorePermission
 .CreateAuthor))
 {
 <abp-button id="NewAuthorButton"
 text="@L["NewAuthor"].Value"
 icon="plus"
 button-type="Primary"/>
 }
 </abp-column>
 </abp-row>
 </abp-card-header>
 <abp-card-body>
 <abp-table striped-rows="true" id="AuthorsTable">
 </abp-card-body>
</abp-card>
```
- This is a simple page similar to the Books page we had created before. It imports a JavaScript file which will be introduced below.
- ## IndexModel.cshtml.cs
- https://docs.abp.io/en/abp/latest/Tutorials/Part-9?UI=MVC&DB=EF
- 2/12

 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 Microsoft.AspNetCore.Mvc.RazorPages;

namespace Acme.BookStore.Web.Pages.Authors
{
 public class IndexModel : PageModel
 {
 public void OnGet()
 {
 }
 }
}
```

## Index.js

 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

```
$(function () {
 var l = abp.localization.getResource('BookStore');
 var createModal = new abp.ModalManager(abp.appPath +
 var editModal = new abp.ModalManager(abp.appPath +

 var dataTable = $('#AuthorsTable').DataTable(
 abp.libs.datatables.normalizeConfiguration({
 serverSide: true,
 paging: true,
 order: [[1, "asc"]],
 searching: false,
 scrollX: true,
 ajax: abp.libs.datatables.createAjax(acme.b
 columnDefs: [
 {
 title: l('Actions'),
 rowAction: {
 items:
 [
 {
 text: l('Edit'),
 visible:
 abp.auth.isGran
 action: function (d
 editModal.open(
 },
 {
 text: l('Delete'),
 visible:
 abp.auth.isGran
 confirmMessage: fun
 return l(
 'AuthorDele
 data.record
);
 },
 action: function (d
 acme.bookStore.
 .delete(dat
 .then(func
 abp.not
 l('
);
 dataTab
));
 }
]
 }
 },
 {
 title: l('Name'),
 data: "name"
 },
 {
 title: l('BirthDate'),
 data: "birthDate",
 render: function (data) {
 return luxon
 .DateTime
```

 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](#)

```
 .fromISO(data, {
 locale: abp.localization
 }).toLocaleString();
 }
 }
]
 })
);

createModal.onResult(function () {
 dataTable.ajax.reload();
});

editModal.onResult(function () {
 dataTable.ajax.reload();
});

$('#NewAuthorButton').click(function (e) {
 e.preventDefault();
 createModal.open();
});
});
```

Briefly, this JavaScript page;

- Creates a Data table with `Actions` , `Name` and `BirthDate` columns.
  - `Actions` column is used to add *Edit* and *Delete* actions.
  - `BirthDate` provides a `render` function to format the `DateTime` value using the [luxon](#) library.
- Uses the `abp.ModalManager` to open *Create* and *Edit* modal forms.

This code is very similar to the Books page created before, so we will not explain it more.

## Localizations

This page uses some localization keys we need to declare. Open the `en.json` file under the `Localization/BookStore` folder of the `Acme.BookStore.Domain.Shared` project and add the following entries:

```
"Menu:Authors": "Authors",
"Authors": "Authors",
"AuthorDeletionConfirmationMessage": "Are you sure to delete this author?",
"BirthDate": "Birth date",
"NewAuthor": "New author"
```

Notice that we've added more keys. They will be used in the next sections.

## Add to the Main Menu

Open the `BookStoreMenuContributor.cs` in the `Menus` folder of the `Acme.BookStore.Web` project and add the following code in the end of the `ConfigureMainMenuAsync` method:

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](#)

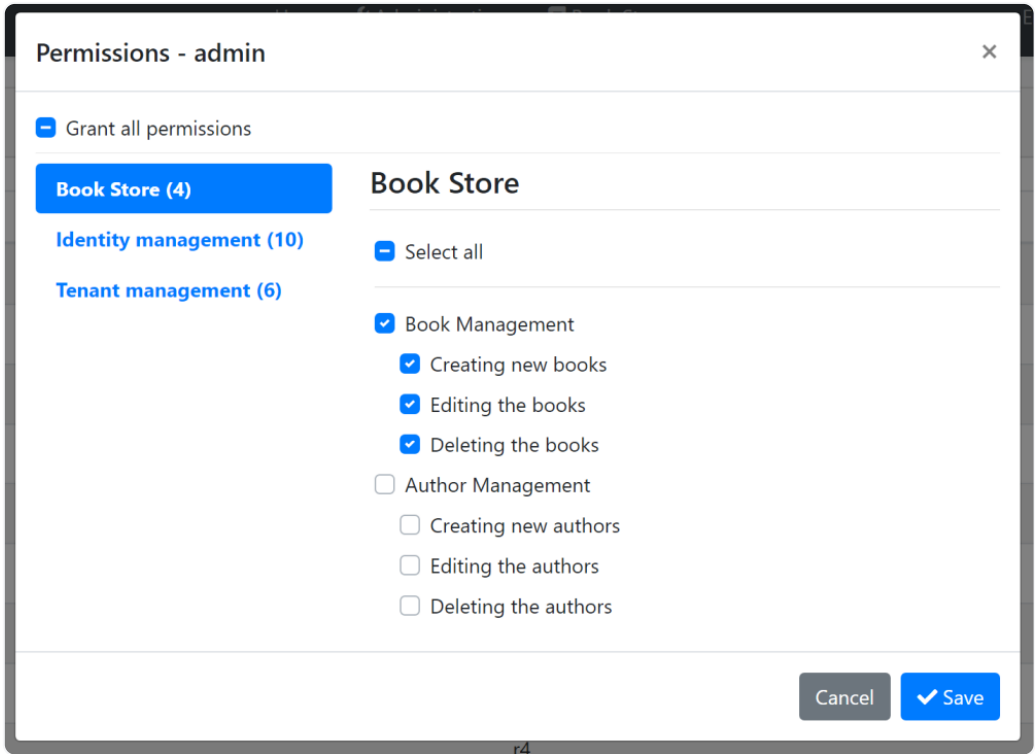
```
if (await context.IsGrantedAsync(BookStorePermissions.A
{
 bookStoreMenu.AddItem(new ApplicationMenuItem(
 "BooksStore.Authors",
 l["Menu:Authors"],
 url: "/Authors"
));
}
```

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

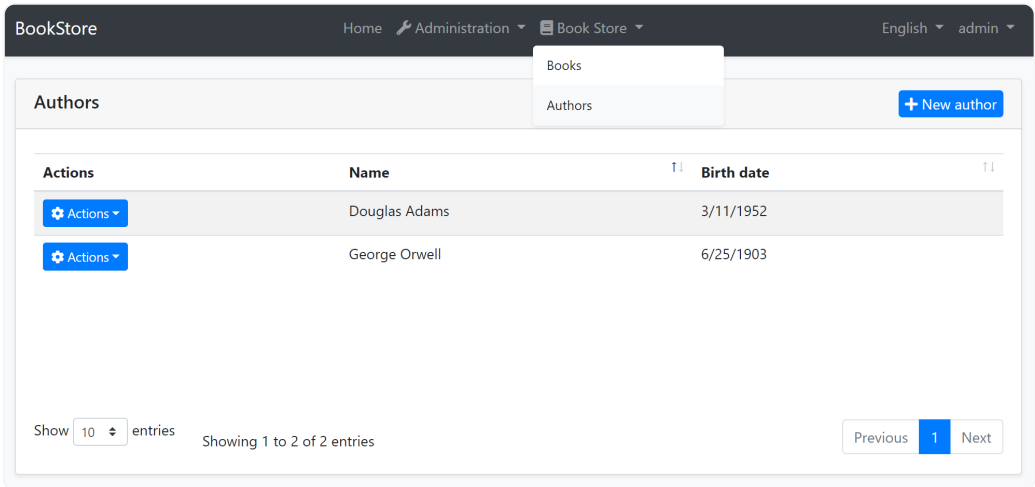
In this document

## Run the Application

Run and login to the application. **You can not see the menu item since you don't have permission yet.** Go to the [Identity/Roles](#) page, click to the *Actions* button and select the *Permissions* action for the **admin role**:



As you see, the admin role has no *Author Management* permissions yet. Click to the checkboxes and save the modal to grant the necessary permissions. You will see the *Authors* menu item under the *Book Store* in the main menu, after **refreshing the page**:



The page is fully working except *New author* and *Actions/Edit* since we haven't implemented them yet.

**Tip:** If you run the [.DbMigrator](#) console application after defining a new permission, it automatically grants these new permissions to the admin role and you don't need to manually grant the permissions yourself.

 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](#)

# Create Modal

Create a new razor page, `CreateModal.cshtml` under the `Pages/Authors` folder of the `Acme.BookStore.Web` project and change the content as given below.

## CreateModal.cshtml

```
@page
@using Acme.BookStore.Localization
@using Acme.BookStore.Web.Pages.Authors
@using Microsoft.Extensions.Localization
@using Vollo.Abp.AspNetCore.Mvc.UI.Bootstrap.TagHelpers
@model CreateModalModel
@Inject IStringLocalizer<BookStoreResource> L
@{
 Layout = null;
}
<form asp-page="/Authors/CreateModal">
 <abp-modal>
 <abp-modal-header title="@L["NewAuthor"].Value">
 <abp-modal-body>
 <abp-input asp-for="Author.Name" />
 <abp-input asp-for="Author.BirthDate" />
 <abp-input asp-for="Author.ShortBio" />
 </abp-modal-body>
 <abp-modal-footer buttons="@((AbpModalButtons.Ca
 </abp-modal>
 </form>
```

We had used [dynamic forms](#) of the ABP Framework for the books page before. We could use the same approach here, but we wanted to show how to do it manually. Actually, not so manually, because we've used `abp-input` tag helper in this case to simplify creating the form elements.

You can definitely use the standard Bootstrap HTML structure, but it requires to write a lot of code. `abp-input` automatically adds validation, localization and other standard elements based on the data type.

## CreateModal.cshtml.cs

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 System;
using System.ComponentModel.DataAnnotations;
using System.Threading.Tasks;
using Acme.BookStore.Authors;
using Microsoft.AspNetCore.Mvc;
using Volo.Abp.AspNetCore.Mvc.UI.Bootstrap.TagHelpers.F

namespace Acme.BookStore.Web.Pages.Authors
{
 public class CreateModalModel : BookStorePageModel
 {
 [BindProperty]
 public CreateAuthorViewModel Author { get; set; }

 private readonly IAuthorAppService _authorAppSe

 public CreateModalModel(IAuthorAppService autho
 {
 _authorAppService = authorAppService;
 }

 public void OnGet()
 {
 Author = new CreateAuthorViewModel();
 }

 public async Task<IActionResult> OnPostAsync()
 {
 var dto = ObjectMapper.Map<CreateAuthorView
 await _authorAppService.CreateAsync(dto);
 return NoContent();
 }

 public class CreateAuthorViewModel
 {
 [Required]
 [StringLength(AuthorConsts.MaxNameLength)]
 public string Name { get; set; }

 [Required]
 [DataType(DataType.Date)]
 public DateTime BirthDate { get; set; }

 [TextArea]
 public string ShortBio { get; set; }
 }
 }
}
```

This page model class simply injects and uses the `IAuthorAppService` to create a new author. The main difference between the book creation model class is that this one is declaring a new class, `CreateAuthorViewModel` , for the view model instead of re-using the `CreateAuthorDto` .

The main reason of this decision was to show you how to use a different model class inside the page. But there is one more benefit: We added two attributes to the class members, which were not present in the `CreateAuthorDto` :



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 `[DataType(DataType.Date)]` attribute to the `BirthDate` which shows a date picker on the UI for this property.
- Added `[TextArea]` attribute to the `ShortBio` which shows a multi-line text area instead of a standard textbox.

In this way, you can specialize the view model class based on your UI requirements without touching to the DTO. As a result of this decision, we have used `ObjectMapper` to map `CreateAuthorViewModel` to `CreateAuthorDto` . To be able to do that, you need to add a new mapping code to the `BookStoreWebAutoMapperProfile` constructor:

```
using Acme.BookStore.Authors; // ADDED NAMESPACE IMPORT
using Acme.BookStore.Books;
using AutoMapper;

namespace Acme.BookStore.Web
{
 public class BookStoreWebAutoMapperProfile : Profile
 {
 public BookStoreWebAutoMapperProfile()
 {
 CreateMap<BookDto, CreateUpdateBookDto>();

 // ADD a NEW MAPPING
 CreateMap<Pages.Authors.CreateModalModel.CreateAuthorDto>();
 }
 }
}
```

"New author" button will work as expected and open a new model when you run the application again:

New author

Name \*

Thomas More

Birth date \*

02.07.1478

ShortBio

Sir Thomas More, venerated in the Catholic Church as Saint Thomas More, was an English lawyer, social philosopher,

Cancel

Save

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

In this document

## Edit 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](#)

Create a new razor page, `EditModal.cshtml` under the `Pages/Authors` folder of the `Acme.BookStore.Web` project and change the content as given below.

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

In this document

## EditModal.cshtml

```
@page
@using Acme.BookStore.Localization
@using Acme.BookStore.Web.Pages.Authors
@using Microsoft.Extensions.Localization
@using Volo.Abp.AspNetCore.Mvc.UI.Bootstrap.TagHelpers
@model EditModalModel
@inject IStringLocalizer<BookStoreResource> L
@{
 Layout = null;
}
<form asp-page="/Authors/EditModal">
 <abp-modal>
 <abp-modal-header title="@L["Update"].Value"></abp-modal-header>
 <abp-modal-body>
 <abp-input asp-for="Author.Id" />
 <abp-input asp-for="Author.Name" />
 <abp-input asp-for="Author.BirthDate" />
 <abp-input asp-for="Author.ShortBio" />
 </abp-modal-body>
 <abp-modal-footer buttons="@((AbpModalButtons.Ca
 </abp-modal>
</form>
```

## EditModal.cshtml.cs

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 System;
using System.ComponentModel.DataAnnotations;
using System.Threading.Tasks;
using Acme.BookStore.Authors;
using Microsoft.AspNetCore.Mvc;
using Volo.Abp.AspNetCore.Mvc.UI.Bootstrap.TagHelpers.F

namespace Acme.BookStore.Web.Pages.Authors
{
 public class EditModalModel : BookStorePageModel
 {
 [BindProperty]
 public EditAuthorViewModel Author { get; set; }

 private readonly IAuthorAppService _authorAppSe

 public EditModalModel(IAuthorAppService authorA
 {
 _authorAppService = authorAppService;
 }

 public async Task OnGetAsync(Guid id)
 {
 var authorDto = await _authorAppService.Get
 Author = ObjectMapper.Map<AuthorDto, EditAu

 }

 public async Task<IActionResult> OnPostAsync()
 {
 await _authorAppService.UpdateAsync(
 Author.Id,
 ObjectMapper.Map<EditAuthorViewModel, U

);

 return NoContent();
 }

 public class EditAuthorViewModel
 {
 [HiddenInput]
 public Guid Id { get; set; }

 [Required]
 [StringLength(AuthorConsts.MaxNameLength)]
 public string Name { get; set; }

 [Required]
 [DataType(DataType.Date)]
 public DateTime BirthDate { get; set; }

 [TextArea]
 public string ShortBio { get; set; }
 }
 }
}
```

This class is similar to the `CreateModal.cshtml.cs` while there are some main differences;

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

- Uses the `IAuthorAppService.GetAsync(...)` method to get the editing author from the application layer.
- `EditAuthorViewModel` has an additional `Id` property which is marked with the `[HiddenInput]` attribute that creates a hidden input for this property.

This class requires to add two object mapping declarations to the `BookStoreWebAutoMapperProfile` class:

```
using Acme.BookStore.Authors;
using Acme.BookStore.Books;
using AutoMapper;

namespace Acme.BookStore.Web
{
 public class BookStoreWebAutoMapperProfile : Profile
 {
 public BookStoreWebAutoMapperProfile()
 {
 CreateMap<BookDto, CreateUpdateBookDto>();

 CreateMap<Pages.Authors.CreateModalModel.CreateAuthorDto>();

 // ADD THESE NEW MAPPINGS
 CreateMap<AuthorDto, Pages.Authors.EditModalModel.EditAuthorDto>();
 CreateMap<Pages.Authors.EditModalModel.EditAuthorDto, AuthorDto>();
 }
 }
}
```

That's all! You can run the application and try to edit an author.

## The Next Part

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

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

In this document