Sports Store: Administration

[Managing Orders 2](#_Toc13413560)

[Enhancing the Model 2](#_Toc13413561)

[Adding the Actions and View 3](#_Toc13413562)

[Adding Catalog Management 5](#_Toc13413563)

# Managing Orders

## Enhancing the Model

1. Adding a property in the Order.cs

using Microsoft.AspNetCore.Mvc.ModelBinding;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

namespace SportStore.Models

{

public class Order

{

[BindNever]

public int OrderID { get; set; }

[BindNever]

public ICollection<CartLine> Lines { get; set; }

**[BindNever]**

**public bool Shipped { get; set; }**

[Required(ErrorMessage = "Please enter a name")]

public string Name { get; set; }

[Required(ErrorMessage = "Please enter the first address line")]

public string Line1 { get; set; }

public string Line2 { get; set; }

public string Line3 { get; set; }

[Required(ErrorMessage = "Please enter a city name")]

public string City { get; set; }

[Required(ErrorMessage = "Please enter a state name")]

public string State { get; set; }

public string Zip { get; set; }

[Required(ErrorMessage = "Please enter a country name")]

public string Country { get; set; }

public bool GiftWrap { get; set; }

}

}

Update the database

dotnet ef migrations add ShippedOrders

## Adding the Actions and View

1. Adding Action methods in the OrderController.cs

using Microsoft.AspNetCore.Mvc;

using SportStore.Models;

using System.Linq;

namespace SportStore.Controllers

{

public class OrderController : Controller

{

private IOrderRepository repository;

private Cart cart;

public OrderController(IOrderRepository repoService, Cart cartService)

{

repository = repoService;

cart = cartService;

}

**public ViewResult List() =>**

**View(repository.Orders.Where(o => !o.Shipped));**

**[HttpPost]**

**public IActionResult MarkShipped(int orderID)**

**{**

**Order order = repository.Orders**

**.FirstOrDefault(o => o.OrderID == orderID);**

**if (order != null)**

**{**

**order.Shipped = true;**

**repository.SaveOrder(order);**

**}**

**return RedirectToAction(nameof(List));**

**}**

public ViewResult Checkout() => View(new Order());

[HttpPost]

public IActionResult Checkout(Order order)

{

if (cart.Lines.Count() == 0)

{

ModelState.AddModelError("", "Sorry, your cart is empty!");

}

if (ModelState.IsValid)

{

order.Lines = cart.Lines.ToArray();

repository.SaveOrder(order);

return RedirectToAction(nameof(Completed));

}

else

{

return View(order);

}

}

public ViewResult Completed()

{

cart.Clear();

return View();

}

}

}

1. Adding a razor view file List.cshtml to the Views/Order folder

@model IEnumerable<Order>

@{

ViewBag.Title = "Orders";

Layout = "\_AdminLayout";

}

@if (Model.Count() > 0)

{

<table class="table table-bordered table-striped">

<tr><th>Name</th><th>Zip</th><th colspan="2">Details</th><th></th></tr>

@foreach (Order o in Model)

{

<tr>

<td>@o.Name</td>

<td>@o.Zip</td>

<th>Product</th>

<th>Quantity</th>

<td>

<**form** **asp-action**="MarkShipped" method="post">

<input type="hidden" name="orderId" value="@o.OrderID" />

<button type="submit" class="btn btn-sm btn-danger">

Ship

</button>

</**form**>

</td>

</tr>

@foreach (CartLine line in o.Lines)

{

<tr>

<td colspan="2"></td>

<td>@line.Product.Name</td>

<td>@line.Quantity</td>

<td></td>

</tr>

}

}

</table>

}

else

{

<div class="text-center">No Unshipped Orders</div>

}

1. Adding Razor Layout file \_AdminLayout.cshtml in the Views/Shared folder

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width" />

<title>@ViewBag.Title</title>

</head>

<body class="m-1 p-1">

<div class="bg-info p-2"><h4>@ViewBag.Title</h4></div>

@RenderBody()

</body>

</html>

# Adding Catalog Management

## Creating a CRUD Controller

1. Adding AdminController.cs

using Microsoft.AspNetCore.Mvc;

using SportStore.Models;

using System.Linq;

namespace SportStore.Controllers

{

public class AdminController : Controller

{

private IProductRepository repository;

public AdminController(IProductRepository repo)

{

repository = repo;

}

public ViewResult Index() => View(repository.Products);

}

}

## Implementing the List View

1. Creating a razor view file Index.cshtml in Views/Admin folder

@model IEnumerable<Product>

@{

ViewBag.Title = "All Products";

Layout = "\_AdminLayout";

}

<table class="table table-striped table-bordered table-sm">

<tr>

<th class="text-right">ID</th>

<th>Name</th>

<th class="text-right">Price</th>

<th class="text-center">Actions</th>

</tr>

@foreach (var item in Model)

{

<tr>

<td class="text-right">@item.ProductID</td>

<td>@item.Name</td>

<td class="text-right">@item.Price.ToString("c")</td>

<td class="text-center">

<**form** **asp-action**="Delete" method="post">

<**a** **asp-action**="Edit" class="btn btn-sm btn-warning"

**asp-route-productId**="@item.ProductID">

Edit

</**a**>

<input type="hidden" name="ProductID" value="@item.ProductID" />

<button type="submit" class="btn btn-danger btn-sm">

Delete

</button>

</**form**>

</td>

</tr>

}

</table>

<div class="text-center">

<**a** **asp-action**="Create" class="btn btn-primary">Add Product</**a**>

</div>

## Editing Products

### Creating the Edit Action Method

1. Updating AdminController.cs

using Microsoft.AspNetCore.Mvc;

using SportStore.Models;

using System.Linq;

namespace SportStore.Controllers

{

public class AdminController : Controller

{

private IProductRepository repository;

public AdminController(IProductRepository repo)

{

repository = repo;

}

public ViewResult Index() => View(repository.Products);

**public ViewResult Edit(int productId) =>**

**View(repository.Products**

**.FirstOrDefault(p => p.ProductID == productId));**

}

}

### Creating the Edit View

1. Creating a Razor view file Edit.cshtml

@model Product

@{

ViewBag.Title = "Edit Product";

Layout = "\_AdminLayout";

}

<**form** **asp-action**="Edit" method="post">

<**input** **type**="hidden" **asp-for**="ProductID" />

<div class="form-group">

<**label** **asp-for**="Name"></**label**>

<**input** **asp-for**="Name" class="form-control" />

</div>

<div class="form-group">

<**label** **asp-for**="Description"></**label**>

<**textarea** **asp-for**="Description" class="form-control"></**textarea**>

</div>

<div class="form-group">

<**label** **asp-for**="Category"></**label**>

<**input** **asp-for**="Category" class="form-control" />

</div>

<div class="form-group">

<**label** **asp-for**="Price"></**label**>

<**input** **asp-for**="Price" class="form-control" />

</div>

<div class="text-center">

<button class="btn btn-primary" type="submit">Save</button>

<**a** **asp-action**="Index" class="btn btn-secondary">Cancel</**a**>

</div>

</**form**>

### Updating the Product Repository

1. Adding method to IProductRepository.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace SportStore.Models

{

public interface IProductRepository

{

//this allows to obtain a sequence of Product objects

IQueryable<Product> Products { get; }

**void SaveProduct(Product product);**

}

}

1. Adding a method to EFProductRepository.cs

using System;

using System.Collections.Generic;

using System.Linq;

namespace SportStore.Models

{

public class EFProductRepository : IProductRepository

{

private ApplicationDbContext context;

public EFProductRepository(ApplicationDbContext ctx)

{

context = ctx;

}

public IQueryable<Product> Products => context.Products;

**public void SaveProduct(Product product)**

**{**

**if (product.ProductID == 0)**

**{**

**context.Products.Add(product);**

**}**

**else**

**{**

**Product dbEntry = context.Products**

**.FirstOrDefault(p => p.ProductID == product.ProductID);**

**if (dbEntry != null)**

**{**

**dbEntry.Name = product.Name;**

**dbEntry.Description = product.Description;**

**dbEntry.Price = product.Price;**

**dbEntry.Category = product.Category;**

**}**

**}**

**context.SaveChanges();**

**}**

}

}

### Handling Edit POST Requests

1. Defining an action method in the AdminController.cs

using Microsoft.AspNetCore.Mvc;

using SportStore.Models;

using System.Linq;

namespace SportStore.Controllers

{

public class AdminController : Controller

{

private IProductRepository repository;

public AdminController(IProductRepository repo)

{

repository = repo;

}

public ViewResult Index() => View(repository.Products);

public ViewResult Edit(int productId) =>

View(repository.Products

.FirstOrDefault(p => p.ProductID == productId));

**[HttpPost]**

**public IActionResult Edit(Product product)**

**{**

**if (ModelState.IsValid)**

**{**

**repository.SaveProduct(product);**

**TempData["message"] = $"{product.Name} has been saved";**

**return RedirectToAction("Index");**

**}**

**else**

**{**

**// there is something wrong with the data values**

**return View(product);**

**}**

**}**

}

}

### Displaying a Confirmation Message

1. Handling ViewBag Message in the \_AdminLayout.cshtml

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width" />

<**link** rel="stylesheet" **asp-href-include**="lib/bootstrap/css/\*.min.css" />

<title>@ViewBag.Title</title>

</head>

<body class="m-1 p-1">

<div class="bg-info p-2"><h4>@ViewBag.Title</h4></div>

**@if (TempData["message"] != null)**

**{**

**<div class="alert alert-success mt-1">@TempData["message"]</div>**

**}**

@RenderBody()

</body>

</html>

### Adding Model Validation

1. Applying Validation attributes in the Product.cs

using System.ComponentModel.DataAnnotations;

namespace SportStore.Models

{

public class Product

{

public int ProductID { get; set; }

[Required(ErrorMessage = "Please enter a product name")]

public string Name { get; set; }

[Required(ErrorMessage = "Please enter a description")]

public string Description { get; set; }

[Required]

[Range(0.01, double.MaxValue,

ErrorMessage = "Please enter a positive price")]

public decimal Price { get; set; }

[Required(ErrorMessage = "Please specify a category")]

public string Category { get; set; }

}

}

1. Adding Validation Error Elements in Edit.cshtml in View/Admin

@model Product

@{

ViewBag.Title = "Edit Product";

Layout = "\_AdminLayout";

}

<**form** **asp-action**="Edit" method="post">

<**input** **type**="hidden" **asp-for**="ProductID" />

<div class="form-group">

<**label** **asp-for**="Name"></**label**>

**<div><span asp-validation-for="Name" class="text-danger"></span></div>**

<**input** **asp-for**="Name" class="form-control" />

</div>

<div class="form-group">

<**label** **asp-for**="Description"></**label**>

**<div><span asp-validation-for="Description" class="text-danger"></span></div>**

<**textarea** **asp-for**="Description" class="form-control"></**textarea**>

</div>

<div class="form-group">

<**label** **asp-for**="Category"></**label**>

**<div><span asp-validation-for="Category" class="text-danger"></span></div>**

<**input** **asp-for**="Category" class="form-control" />

</div>

<div class="form-group">

<**label** **asp-for**="Price"></**label**>

**<div><span asp-validation-for="Price" class="text-danger"></span></div>**

<**input** **asp-for**="Price" class="form-control" />

</div>

<div class="text-center">

<button class="btn btn-primary" type="submit">Save</button>

<**a** **asp-action**="Index" class="btn btn-secondary">Cancel</**a**>

</div>

</**form**>

1. Adding CSS to \_AdminLayout.cshtml in View/Shared Folder

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width" />

<**link** rel="stylesheet" **asp-href-include**="lib/bootstrap/css/\*.min.css" />

<title>@ViewBag.Title</title>

**<style>**

**.input-validation-error {**

**border-color: red;**

**background-color: #fee;**

**}**

**</style>**

</head>

<body class="m-1 p-1">

<div class="bg-info p-2"><h4>@ViewBag.Title</h4></div>

@if (TempData["message"] != null)

{

<div class="alert alert-success mt-1">@TempData["message"]</div>

}

@RenderBody()

</body>

</html>

### Enabling Client-side Validation

1. Adding JavaScript package in libman.json

{

"version": "1.0",

"defaultProvider": "cdnjs",

"libraries": [

{

"library": "twitter-bootstrap@4.0.0-alpha.6",

"destination": "wwwroot/lib/bootstrap"

},

{

"library": "font-awesome@5.9.0",

"destination": "wwwroot/lib/fontawesome"

},

**{**

**"library": "jquery@3.4.1",**

**"destination": "wwwroot/lib/jquery"**

**},**

**{**

**"library": "jquery-validate@1.19.1",**

**"destination": "wwwroot/lib/jquery-validate"**

**},**

**{**

**"library": "jquery-validation-unobtrusive@3.2.11",**

**"destination": "wwwroot/lib/jquery-validation-unobtrusive"**

**}**

]

}

1. Adding validation libraries to the \_AdminLayout.cshtml in View/Shared folder

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width" />

<**link** rel="stylesheet" **asp-href-include**="lib/bootstrap/css/\*.min.css" />

<title>@ViewBag.Title</title>

<style>

.input-validation-error {

border-color: red;

background-color: #fee;

}

</style>

**<script src="~/lib/jquery/jquery.min.js"></script>**

**<script src="~/lib/jquery-validate/jquery.validate.min.js"></script>**

**<script src="~/lib/jquery-validation-unobtrusive/jquery.validate.unobtrusive.min.js">**

</script>

</head>

<body class="m-1 p-1">

<div class="bg-info p-2"><h4>@ViewBag.Title</h4></div>

@if (TempData["message"] != null)

{

<div class="alert alert-success mt-1">@TempData["message"]</div>

}

@RenderBody()

</body>

</html>

## Creating new products

1. Adding Create Action to AdminController.cs

using Microsoft.AspNetCore.Mvc;

using SportStore.Models;

using System.Linq;

namespace SportStore.Controllers

{

public class AdminController : Controller

{

private IProductRepository repository;

public AdminController(IProductRepository repo)

{

repository = repo;

}

public ViewResult Index() => View(repository.Products);

public ViewResult Edit(int productId) =>

View(repository.Products

.FirstOrDefault(p => p.ProductID == productId));

[HttpPost]

public IActionResult Edit(Product product)

{

if (ModelState.IsValid)

{

repository.SaveProduct(product);

TempData["message"] = $"{product.Name} has been saved";

return RedirectToAction("Index");

}

else

{

// there is something wrong with the data values

return View(product);

}

}

**public ViewResult Create() => View("Edit", new Product());**

}

}

## Deleting products

1. Adding a Method to the IProductRepository.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace SportStore.Models

{

public interface IProductRepository

{

//this allows to obtain a sequence of Product objects

IQueryable<Product> Products { get; }

void SaveProduct(Product product);

**Product DeleteProduct(int ProductID);**

}

}

1. Implementing Deletion support in the EFProductRepository.cs

using System;

using System.Collections.Generic;

using System.Linq;

namespace SportStore.Models

{

public class EFProductRepository : IProductRepository

{

…

**public Product DeleteProduct(int productID)**

**{**

**Product dbEntry = context.Products**

**.FirstOrDefault(p => p.ProductID == productID);**

**if (dbEntry != null)**

**{**

**context.Products.Remove(dbEntry);**

**context.SaveChanges();**

**}**

**return dbEntry;**

**}**

}

}

1. Adding Deletion action method in the AdminController.cs

using Microsoft.AspNetCore.Mvc;

using SportStore.Models;

using System.Linq;

namespace SportStore.Controllers

{

public class AdminController : Controller

{

private IProductRepository repository;

public AdminController(IProductRepository repo)

{

repository = repo;

}

public ViewResult Index() => View(repository.Products);

public ViewResult Edit(int productId) =>

View(repository.Products

.FirstOrDefault(p => p.ProductID == productId));

[HttpPost]

public IActionResult Edit(Product product)

{

if (ModelState.IsValid)

{

repository.SaveProduct(product);

TempData["message"] = $"{product.Name} has been saved";

return RedirectToAction("Index");

}

else

{

// there is something wrong with the data values

return View(product);

}

}

public ViewResult Create() => View("Edit", new Product());

**[HttpPost]**

**public IActionResult Delete(int productId)**

**{**

**Product deletedProduct = repository.DeleteProduct(productId);**

**if (deletedProduct != null)**

**{**

**TempData["message"] = $"{deletedProduct.Name} was deleted";**

**}**

**return RedirectToAction("Index");**

**}**

}

}

# Screenshot













