**Работа в клас**

**MVC (Model-View-Controller)**

е архитектурен шаблон, който разделя приложението на три основни компонента:

**Модел (Model) – отговаря за данните**

* Моделът управлява данните на приложението. Това включва структурирането,

валидирането и бизнес логиката.

* Моделът често е свързан с ORM (ObjectRelational Mapping) фреймуърк като

Entity Framework, който помага за взаимодействие с базата данни.

* Класове като Product, User и др., са представени като таблици в базата данни

**Изглед (View) – отговаря за визуализацията на данните**

* Изгледът показва данните на потребителите. Той съдържа HTML, CSS и  JavaScript.
* Може да генерира динамично съдържание, като използва  данни от модела.
* Ние използваме Razor, който визуализира HTML кода с данни от модела.

**Контролер (Controller) – управлява взаимодействието между потребителя и модела**

* Контролерът приема потребителските заявки, обработва ги и връща подходящи

отговори.

* Контролерът определя кой изглед да се покаже и с какви данни.
* Методите в контролерите, които обработват заявки, се наричат действия (action).

**Как работят заедно?**

1. Потребителят прави заявка: Например, заявка за показване на всички продукти.
2. Контролерът я обработва: Контролерът получава заявката, комуникира с модела, за да получи нужните данни.
3. Моделът връща списък с продукти.
4. Контролерът избира подходящия изглед и му предава данните.
5. Изгледът генерира HTML, който се изпраща обратно към потребителя.

**Конвенция за именуване на модели, изгледи и контролери. Ето някои добри практики:**

**Модели**

* Име на клас: Единствено число, например **Product, User, Order**.
* Име на файл: Същото като името на класа. Например, **Product**.cs.

**Изгледи**

* Име на файла съвпада с името на действието в контролера. Например, ако действието е **Index()**, изгледът ще се казва **Index**.cshtml.
* Местоположение: В папка, наречена след контролера. Например, изгледите за

**ProductController** се намират в Views/**Product**.

**Контролери**

* Името на контролера следва да завършва с "Controller".

Например:  ProductController, UserController.

* А името на файла е същото като името на класа. Например: ProductController.cs.

**Пример**

За ProductController:

* Модел: Product.cs
* Изгледи: В папка Views/Product/ с файлове Index.cshtml, Edit.cshtml, и т.н.
* Контролер: ProductController.cs

**Следването на тези конвенции помага на екипите да се ориентират добре в кода и**

**улеснява работата на IoC контейнера и Entity-Framework-a.**

1. **Controllers:**

public class ProductController : Controller

{

private readonly IEnumerable<ProductViewModel> product = new List<ProductViewModel>()

{

new ProductViewModel()

{

Id = 1,

Name = "Cheese",

Price=2.34

},

new ProductViewModel()

{

Id = 2,

Name = "Ham",

Price= 3.59

},

new ProductViewModel()

{

Id = 3,

Name = "Bread",

Price= 6.59

}

};

public IActionResult Index()

{

return View(product);

}

public IActionResult ById(int id)

{

if(id < 1 || id > product.Count())

{

TempData["Error"] = "No such product";

return RedirectToAction(nameof(Index));

}

return View(product.FirstOrDefault(p => p.Id == id));

}

}

public class NumbersController : Controller

{

[HttpGet]

public IActionResult Index()

{

return View();

}

[HttpPost]

public IActionResult Limit(RangeViewModel model)

model.Start = 1;

return View("Index", model);

}

}

public class AccountController : Controller

{

[HttpGet]

public IActionResult Login()

{

return View();

}

[HttpPost]

public IActionResult Login(LoginFormViewModel model)

{

if(model.Email == "admin" && model.Password == "123")

{

return RedirectToAction("Index", "Book");

}

else

{

model.LoginMessage = "Invalid Login and password combination.";

}

return View(model);

}

}

1. **Models:**

public class ProductViewModel

{

public int Id { get; set; }

public required string Name { get; set; }

public double Price { get; set; }

}

namespace AspNetCore.Models.NumbersViewModels

{

public class RangeViewModel

{

public int Start { get; set; }

public int End { get; set; }

}

}

public class LoginFormViewModel

{

public string Email { get; set; } = string.Empty;

public string Password { get; set; }= string.Empty;

public string LoginMessage { get; set; } = string.Empty;

}

1. **Views:**
   1. **Product**
      1. **Product/Index.cshtml**

@model IEnumerable<ProductViewModel>

@{

ViewBag.Title = "All Products";

}

<h2>@ViewBag.Title</h2>

<span class="text-danger">@TempData["Error"]</span>

<div class="row">

<div class="col-5">

<div class="card">

<div class="card-header">@ViewBag.Title</div>

<ul class="list-group list-group-flush">

@foreach(var item in Model)

{

<li class="list-group-item">

<**a** **asp-action**="ById" **asp-route-id**="@item.Id">

N.@item.Id: @item.Name - @item.Price lv.

</**a**>

</li>

}

</ul>

</div>

</div>

</div>

* + 1. **Product/ById.cshtml**

@model ProductViewModel

@{

ViewData["Title"] = $"Product {Model.Name}";

}

<h2>@ViewData["Title"]</h2>

<div class="row">

<div class="col-5">

<div class="card">

<div class="card-header">Product N. @Model.Id</div>

<ul class="list-group list-group-flush">

<li class="list-group-item">

<h3>Product Name: @Model.Name</h3>

Price: @Model.Price lv.

</li>

</ul>

</div>

</div>

</div>

* 1. **Numbers/Index.cshtml**

@model AspNetCore.Models.NumbersViewModels.RangeViewModel

@{

if (Model == null)

{

ViewData["Title"] = $"Choose range";

}

else

ViewData["Title"] = $"First {Model.End} Numbers";

}

<h1>@ViewData["Title"]</h1>

@if(Model != null)

{

<ul>

@for (int i = Model.Start; i <= Model.End; i++)

{

<li>@i</li>

}

</ul>

}

<**form** method="post" **asp-controller**="Numbers" **asp-action**="Limit" >

<**input** **asp-for**="@Model.End" **type**="number" />

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

</**form**>

* 1. **Account/Login.cshtml**

@model AspNetCore.Models.Account.LoginFormViewModel

@if(Model != null)

{

<p>@Model.LoginMessage</p>

}

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

<div class="form-group">

<**label** **asp-for**="@Model.Email">Email address</**label**>

<**input** class="form-control" **asp-for**="@Model.Email" aria-describedby="emailHelp" placeholder="Enter email">

<small id="emailHelp" class="form-text text-muted">We'll never share your email with anyone else.</small>

</div>

<div class="form-group">

<**label** **asp-for**="@Model.Password">Password</**label**>

<**input** **type**="password" class="form-control" **asp-for**="@Model.Password" placeholder="Password">

</div>

<div class="form-group form-check">

<input type="checkbox" class="form-check-input" id="exampleCheck1">

<label class="form-check-label" for="exampleCheck1">Check me out</label>

</div>

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

</**form**>