

ASP.NET Active Dynamic Web Pages

השלמות משיעור קודם בנושא CORS Middleware והגשת קבצים סטטיים

מבוא והכרות עם ASP.Net Core MVC
מבנה הפרוייקט,
Controllers, Views, Actions

Middleware

לשליחת CORS Headers

פתרון מובנה עם יותר יכולות ואפשרויות - כגון ציון headers/cookies שמותר ללקוח לשלוח או שאסור לו לשלוח

```
var AllowedOrigins = "AllowedOrigins";
```

1

```
builder.Services.AddCors(options =>
{
    options.AddPolicy(name: AllowedOrigins,
        policy =>
        {
            policy.WithOrigins("http://localhost:5173",
                              "http://localhost:3000");
            policy.AllowAnyHeader();
            policy.AllowAnyMethod();
            policy.AllowCredentials();
        });
});
```

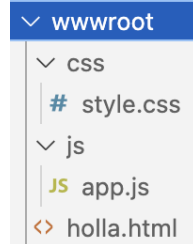
2

```
var app = builder.Build();
```

```
app.UseCors(AllowedOrigins);
```

3

Middleware: להגשת קבצים סטטיים



1 ניצור תיקיה בשם - wwwroot שם התיקיה חשוב והוא לפי סטנדרט

```
var app = builder.Build();
app.UseStaticFiles();
app.UseCors(AllowedOrigins);
```

2 נוסיף את השורה הבאה לקובץ Program.cs:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Holla</title>
  <link rel="stylesheet" href="css/style.css" />
  <script defer src="./js/app.js"></script>
</head>
<body>
  <h1 id="typewriter">Holla</h1>
</body>
</html>
```

3 נכניס קבצים לתיקיה ונריץ מחדש את הפרוייקט:

נבקר בדפדפן בכתובת:
<http://localhost:5059/holla.htm>
ונקבל את הקובץ שלנו

4

ASP.NET Core MVC

פרויקט חדש – סוג הפרוייקט:



ASP.NET Core Web App (Model-View-Controller)

A project template for creating an ASP.NET Core application with example ASP.NET Core MVC Views and Controllers. This template can also be used for RESTful HTTP services.

C#

Linux

macOS

Windows

Cloud

Service

Web

ASP.NET Core MVC

פרויקט חדש – סוג הפרוייקט:

Additional information

ASP.NET Core Web App (Model-View-Controller)

C#

Linux

macOS

Windows

Cloud

Service

Web

Framework [i](#)

.NET 8.0 (Long Term Support)

Authentication type [i](#)

None

נוסיף את זה בהמשך – כרגע לא צריך

☒ Configure for HTTPS [i](#)

☐ Enable container support [i](#)

Container QS [i](#)

Linux

Container build type [i](#)

Dockerfile

☒ Do not use top-level statements [i](#)

☐ Enlist in .NET Aspire orchestration [i](#)

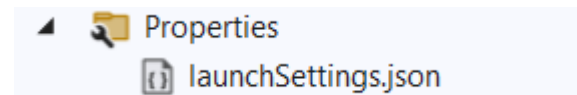
ASP.NET Core MVC

מבנה הפרוייקט

launchsettings.json

hemastore.org/launchsettings.json

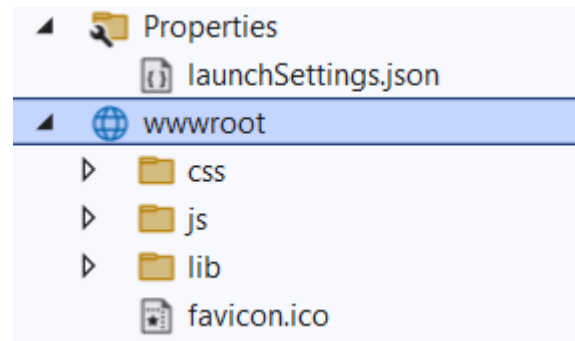
```
{,
  "profiles": {
    "http": {
      "commandName": "Project",
      "dotnetRunMessages": true,
      "launchBrowser": true,
      "applicationUrl": "http://localhost:5096",
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
      }
    }
  },
```



הגדרת משתני סביבה

ASP.NET Core MVC

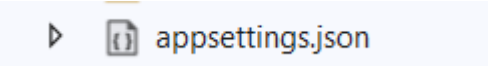
מבנה הפרוייקט



תיקיה לתוכן סטטי:
קבצים סטטיים:
תמונות, עיצוב וכו'

ASP.NET Core MVC

מבנה הפרוייקט



appsettings.json

כאן נשמור חיבורים לדטה-בייס וכו'

ASP.NET Core MVC

מבנה הפרוייקט

```
✓ app.MapControllerRoute(  
    |     name: "default",  
    |     pattern: "{controller=Home}/{action=Index}/{id?}");
```

תבנית לניווט:

baseUrl/

ינווט אותנו לקונטרולר הבית Home

במתודה Index

בלי Id

תבנית לניווט:

baseUrl/about/index/1

ינווט אותנו לקונטרולר של Books

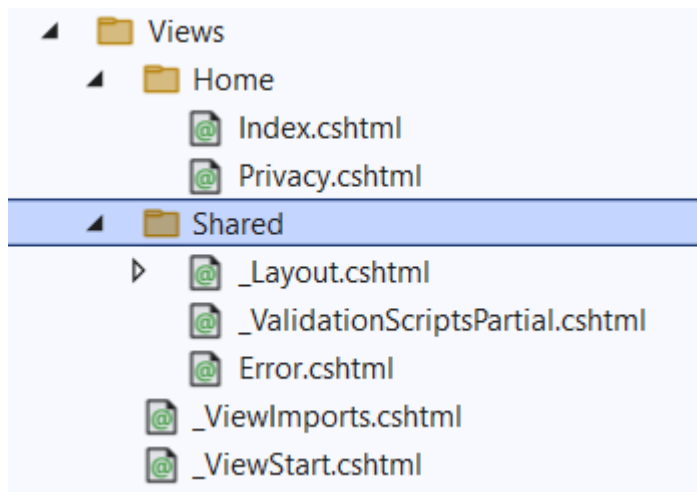
במתודה (Action) בשם Index

ואם כתוב 1 – אז יעביר בפרמטר 1

ASP.NET Core MVC

מבנה הפרוייקט

תיקית Views:



שם התיקיה מזהה אוטומטית. בתוכה יש לנו את ה Views שקרויים על שם ה Actions שהם משרתים ב Controller של Home

הקובץ _ViewStart הוא נקודת אתחול לכל View והוא אחראי על קביעת ה Layout והוא מחפש את הקבצים בתיקיה **Views/Shared** אז שם התיקיה חשוב

_ViewStart.cshtml

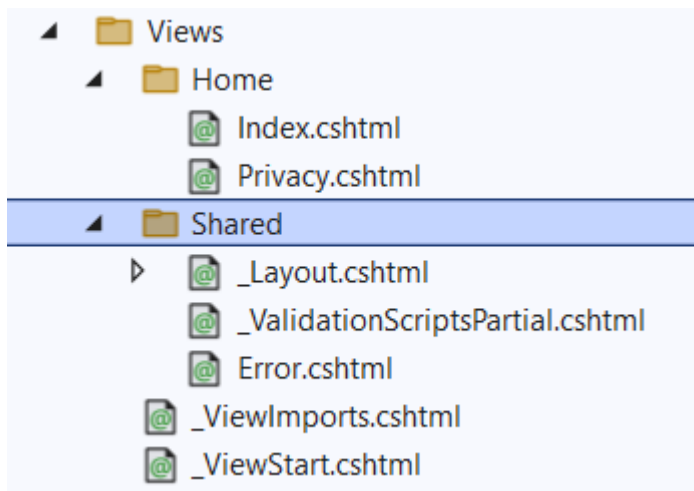
```
@{  
    Layout = "_Layout";  
}
```

הקובץ _ViewImports.cshtml הוא קובץ ל using שאנחנו צריכים בכל ה Views.

```
✓ @using FSMVC  
  @using FSMVC.Models  
  @addTagHelper *, Microsoft.AspNetCore.Mvc.TagHelpers
```

ASP.NET Core MVC

מבנה הפרוייקט



הקובץ _Layout.cshtml

הוא קובץ שמוסיף לנו Header Footer ואיזור תוכן שלתוכו ירונדר ה View

_Layout.cshtml

```
</header>
<div class="container">
  <main role="main" class="pb-3">
    @RenderBody()
  </main>
</div>
```

```
<footer class="border-top footer text-muted">
```

הקובץ _ValidationPartials.cshtml

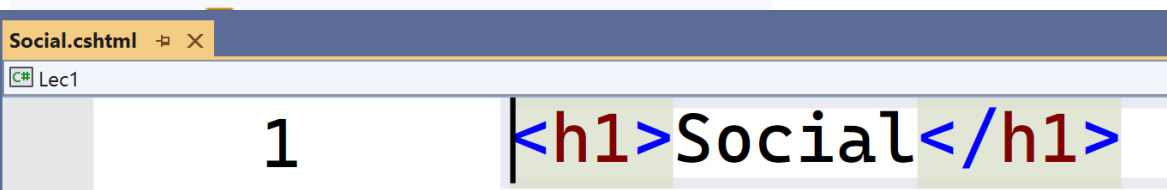
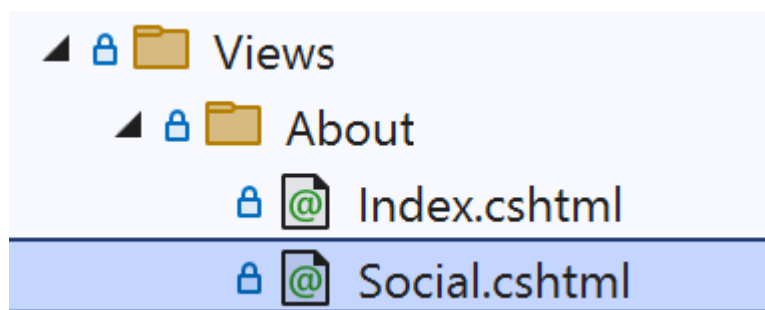
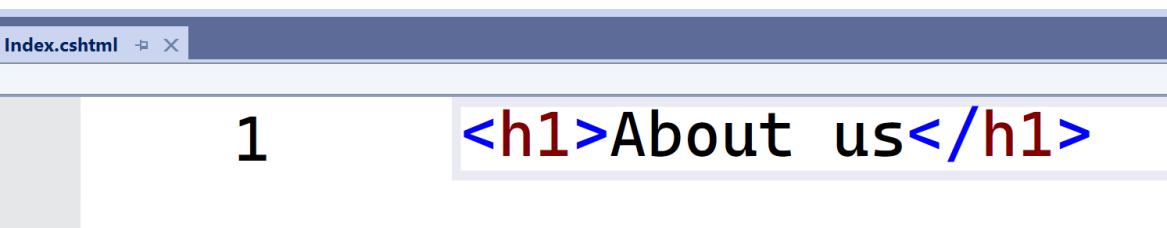
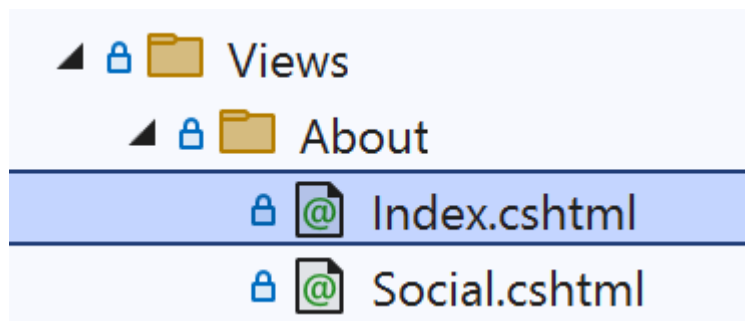
מכיל using שאנחנו צריכים רק חלק מהדפים.

About חדש להצגת Controller

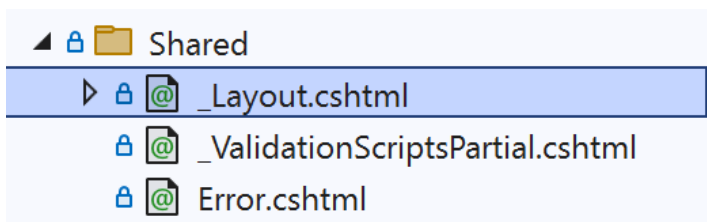
```
public class AboutController(ILogger<AboutController> logger) : Controller
{
    0 references
    public IActionResult Index(int? id)
    {
        logger.LogInformation(id is null ? "No id" : $"id: {id}");
        return View();
    }

    // localhost:3333/About/Social
    0 references
    public IActionResult Social()
    {
        return View();
    }
}
```

About להצגת Views



הוספת קישורים לסרגל הניווט:



```
<div class="navbar-collapse collapse d-sm-inline-flex justify-content-between">
  <ul class="navbar-nav flex-grow-1">
    <li class="nav-item">
      <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Index">Home</a>
    </li>
    <li class="nav-item">
      <a class="nav-link text-dark" asp-area="" asp-controller="Home" asp-action="Privacy">Privacy</a>
    </li>
    <li class="nav-item">
      <a class="nav-link text-dark" asp-controller="About" asp-action="Index">About Us</a>
    </li>
    <li class="nav-item">
      <a class="nav-link text-dark" asp-controller="About" asp-action="Social">Social</a>
    </li>
  </ul>
</div>
```

ASP.NET Core MVC

מודל חדש – BookViewModel

```
namespace Lec1.Models;|
```

9 references

```
✓ public class BookViewModel
```

```
{
```

4 references

```
public int Id { get; set; }
```

5 references

```
public string? Title { get; set; }
```

4 references

```
public string? Description { get; set; }
```

```
}
```

Controller חדש להצגת מידע על ספרים:

```
public class BooksController : Controller
{
    private static List<BookViewModel> books = [
        new BookViewModel() {
            Id = 1,
            Title = "Harry potter 1",
            Description = "The only book :-)"
        },
        new BookViewModel() {
            Id = 2,
            Title = "The Godfather",
            Description = "The only other book :-)"
        }
    ];
    0 references
    public IActionResult Index()
    {
        return View(books);
    }
}
```

העברה של אובייקט לView

View להצגת רשימה של ספרים:

```
@model List<BookViewModel>
```

```
<h1>Books Page</h1>
```

```
<table class="table">
```

```
<thead>
```

```
<tr>
```

```
<th>Title</th>
```

```
<th>Description</th>
```

```
</tr>
```

```
</thead>
```

```
<tbody>
```

```
@foreach (var item in Model)
```

```
{
```

```
<tr>
```

```
<td>@item.Title</td>
```

```
<td>@item.Description</td>
```

```
</tr>
```

```
}
```

```
</tbody>
```

```
</table>
```

הכרזה על הטיפוס של
המודל

לולאת @foreach

סינטקס של רייוזור
להצגה של תכונות
מאובייקט:



תרגיל:

```
@model List<BookViewModel>
```

```
<h1>Books Page</h1>
```

```
<table class="table">
```

```
  <thead>
```

```
    <tr>
```

```
      <th>Title</th>
```

```
      <th>Description</th>
```

```
    </tr>
```

```
  </thead>
```

```
  <tbody>
```

```
    @foreach (var item in Model)
```

```
    {
```

```
      <tr>
```

```
        <td>@item.Title</td>
```

```
        <td>@item.Description</td>
```

```
      </tr>
```

```
    }
```

```
  </tbody>
```

```
</table>
```

בלחיצה על השורה נעבור לעמוד אחר:

BOOKS/DETAILS/1

תרגיל: צרו מתודה בשם
BooksController details
המתודה תקבל int id כפרמטר

יש ליצור View חדש בתיקית Views של Books עבור הDetails

```
@model List<BookViewModel>
<h1>Books Page</h1>

<table class="table">
  <thead>
    <tr>
      <th>Title</th>
      <th>Description</th>
    </tr>
  </thead>
  <tbody>
    @foreach (var item in Model)
    {
      <tr>
        <td>
          <a class="text-dark text-decoration-none" href="/books/details/@item.Id">
            @item.Title
          </a>
        </td>
        <td>@item.Description</td>
      </tr>
    }
  </tbody>
</table>
```

פתרון לתרגיל הקודם

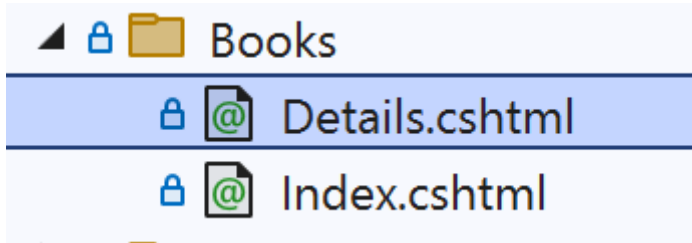


פתרון לתרגיל:

```
public class BooksController : Controller
{
    private static List<BookViewModel> books = [
        new BookViewModel() {
            Id = 1,
            Title = "Harry potter 1",
            Description = "The only book :-)"
        },
        new BookViewModel() {
            Id = 2,
            Title = "The Godfather",
            Description = "The only other book :-)"
        }
    ];
    0 references
    public IActionResult Index()
    {
        return View(books);
    }

    0 references
    public IActionResult Details(int id)
    {
        var book = books.FirstOrDefault(b => b.Id == id);
        return View(book);
    }
}
```

פתרון לתרגיל:



```
@model BookViewModel
```

```
<h1>@Model.Title Details</h1>
```

Title:

```
@Model.Title
```

Description:

```
@Model.Description
```

שיעורי בית:

- (1) צרו פרוייקט MVC חדש בשם MoviesProject
- (2) צרו Controller להצגת מידע על סרטים
- (3) צרו מודל לסרט – תכונות Id, title, overview, imageUrl
- (4) צרו עמוד להצגת כל הסרטים
- (5) בלחיצה על סרט יש לעבור לעמוד פרטי הסרט ולהציג את הסרט בעמוד בודד.
- (6) הוסיפו את ה Controller שלכם לסרגל הניווט