

תכנות באינטרנט

ASP.Net Core

Razor pages

גלעד מרקמן

קריית החינוך פארק המדע,

נס ציונה



ASP.NET Core



ASP.Net Core vs net framework

- טכנולוגיה חדשה המחליפה את .net framework
- קוד פתוח
- ניתנת להרצה על windows, mac, linux

.NET Core

*Cross-platform, modular libraries & runtime
optimized for server and cloud workloads*

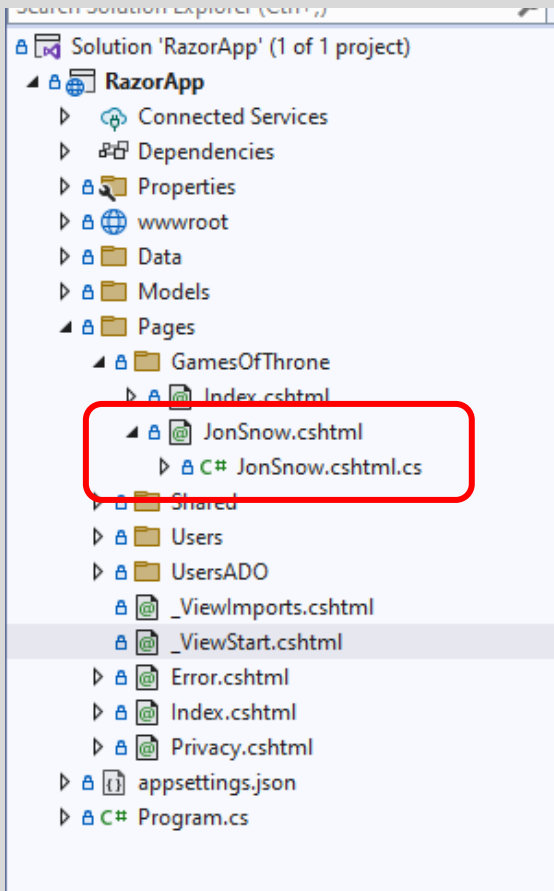


Razor Pages

<https://www.youtube.com/watch?v=6GRFDkeCv3k&t=42s>

<https://www.youtube.com/watch?v=68towqYcQlY&t=266s>

<https://www.youtube.com/watch?v=eru2emiqow0>

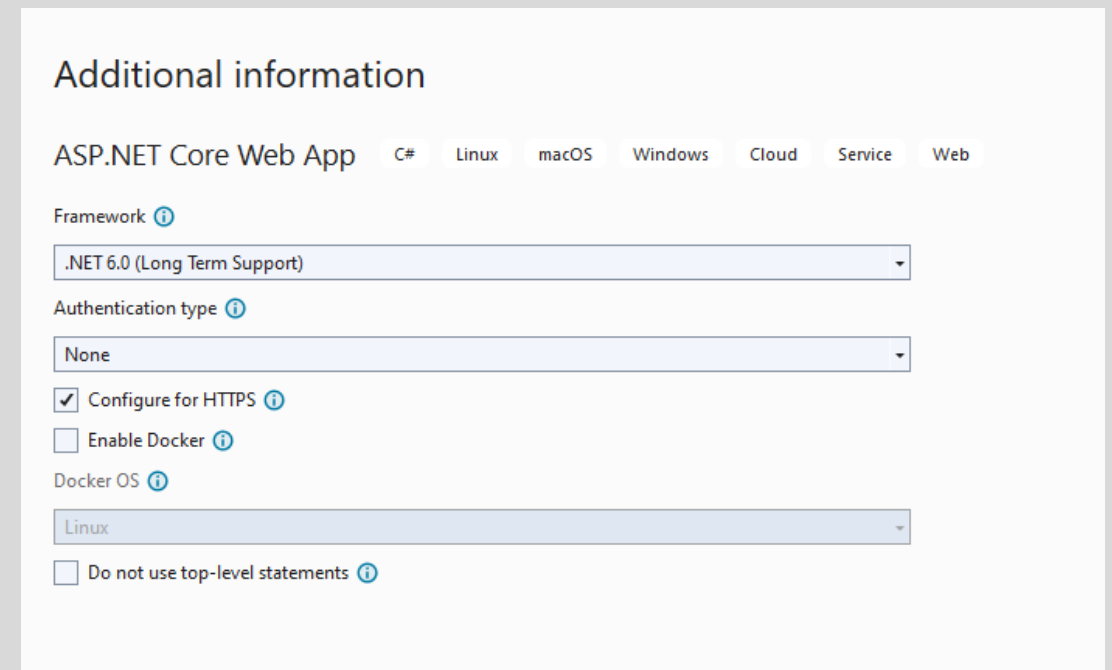
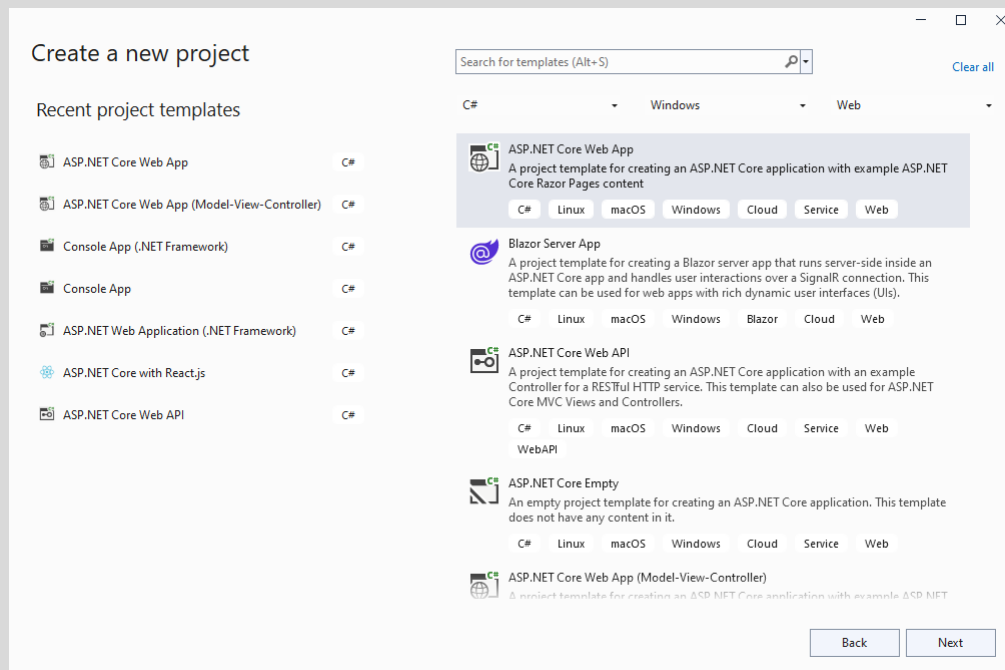


- Razor Pages דומים ל web forms הם כוללים דף HTML ואת הקוד מאחורי הדף בקובץ נפרד.

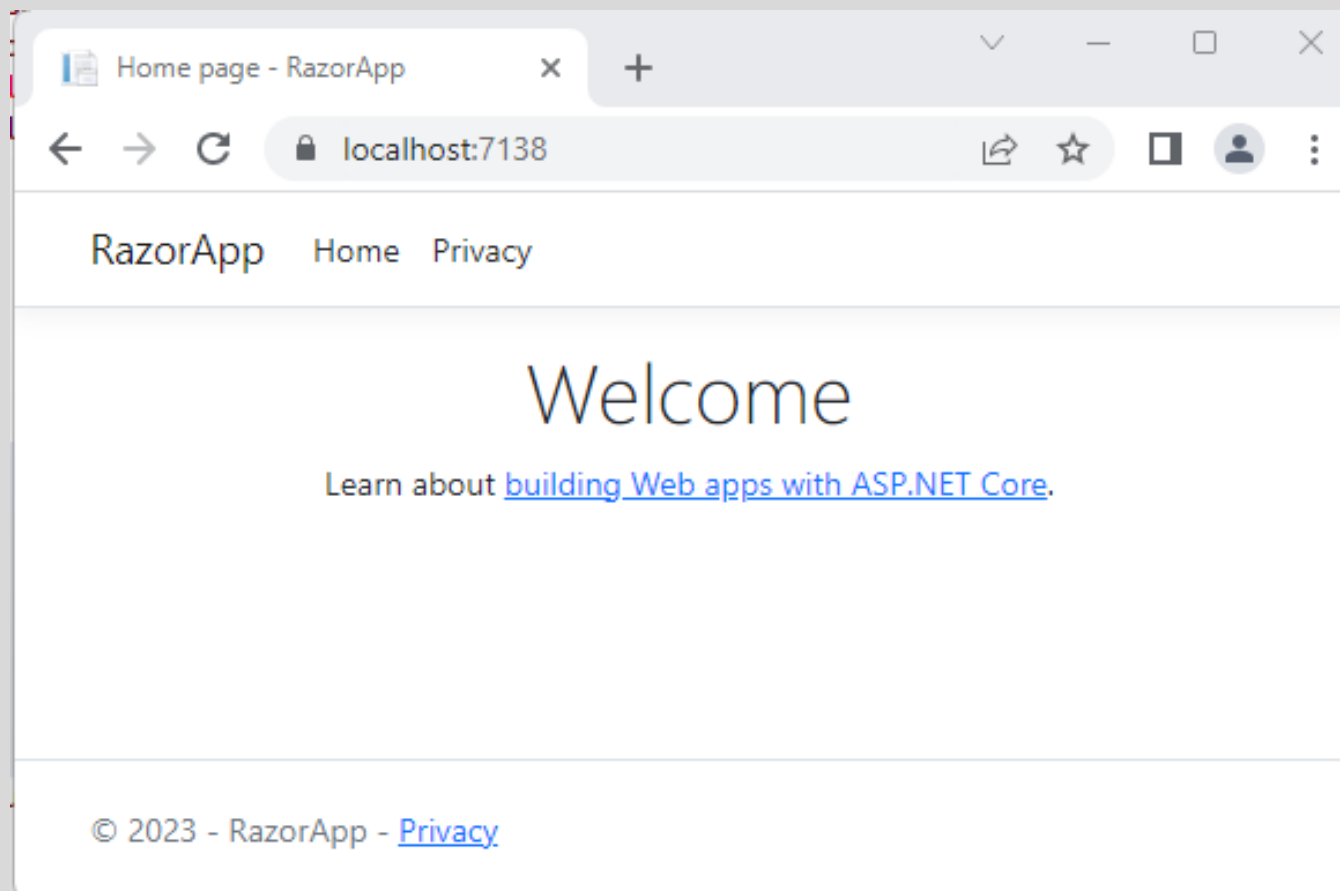
- קובץ ה view כולל תגיות HTML וכן ניתן להוסיף קוד ב C#. הסיומת היא cshtml.

- קובץ הקוד צמוד לקובץ view והוא כולו נכתב ב C#. הסיומת היא cshtml.cs.

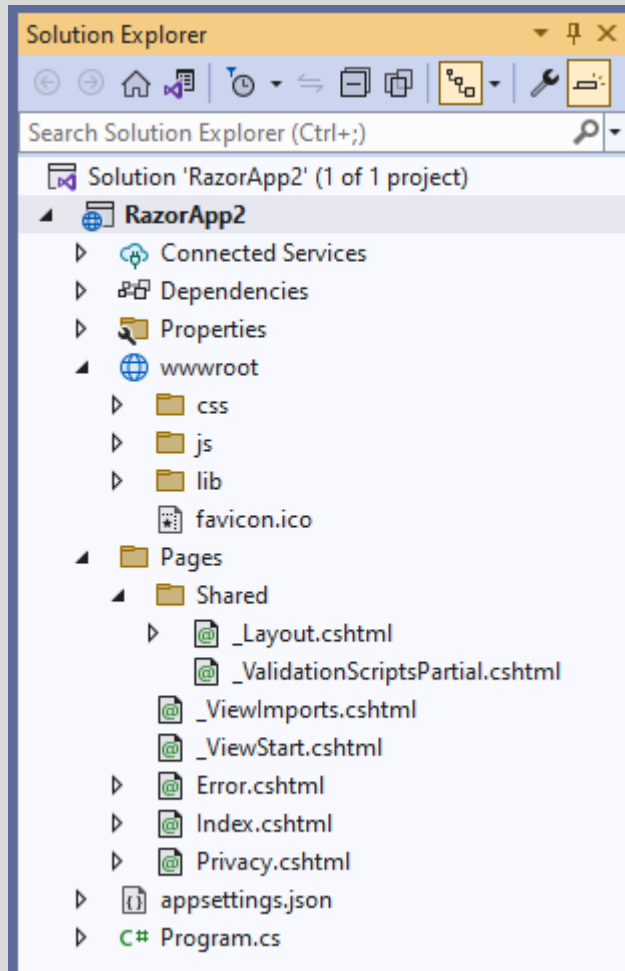
- ביצירת פרויקט חדש נבחר ASP.Net Core Web App.
- נקבע שם לפרויקט ומקום לשמור אותו.
- לאחר מכן נבחר פרטים נוספים (כגון: האם להוסיף Authentication)



הפרויקט החדש



מבנה הפרויקט



- תוכנית ראשית Program.cs.
- appsettings.json
- wwwroot – קבצים סטטיים (כגון js, css, html). כולל ספריית bootstrap לעיצוב.
- Pages – הכולל את דפי ה Razor pages
 - תבנית ראשית _Layout (מסטר פייג).
 - Index.cshtml
 - Privacy.cshtml
- בהמשך נבנה ספריית Models ו-Data.

- ספריית Pages
 - index.cshtml
 - Privacy.cshtml

- קובץ מסטר פייג
 - Shared/_Layout.cshtml

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>@ViewData["Title"] - CoreAppNoAuth</title>
  <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
  <link rel="stylesheet" href="~/css/site.css" asp-append-version="true" />
  <link rel="stylesheet" href="~/CoreAppNoAuth.styles.css" asp-append-version="true" />
</head>
<body>
  <header>...
  <div class="container">
    <main role="main" class="pb-3">
      @RenderBody()
    </main>
  </div>

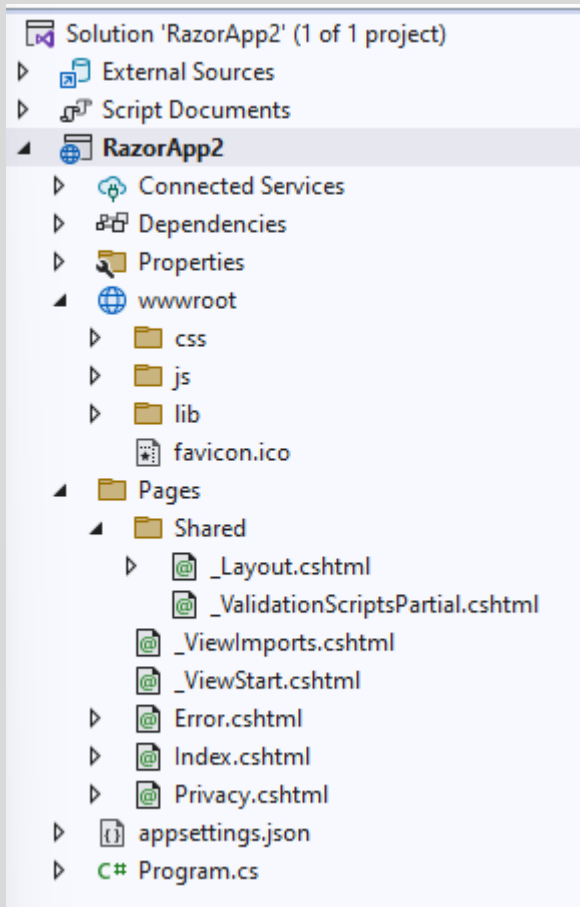
  <footer class="border-top footer text-muted">...
  <script src="~/lib/jquery/dist/jquery.min.js"></script>
  <script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
  <script src="~/js/site.js" asp-append-version="true"></script>
  @await RenderSectionAsync("Scripts", required: false)
</body>
</html>
```


- בפרויקט Razor Pages הרוטינג מתבצע בהתאם למיקום הפיסי של הקבצים בפרויקט.

- המיקום ההתחלתי הוא pages, ולכן פניה לדף Privacy תעשה על ידי: `localhost:7120/Privacy`

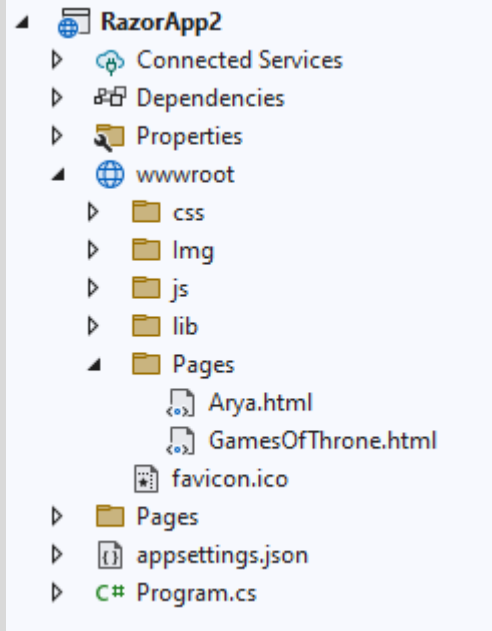
- הדף ההתחלתי הוא index הנמצא בתוך Pages. אם לא מציינים שם דף הפניה היא ל index.

- אם ניצור תת ספריה בתוך Pages הפניה אל דפים הנמצאים בה תהיה: `https://localhost:7120/folder/pageName`



`app.UseStaticFiles();`

- ניתן להציג דפי HTML אם ב program קיימת הפקודה:
- הוספת דפי HTML תעשה רק בתוך הספרייה `wwwroot`.
- נפתח ספרייה חדשה בשם Pages ונוסיף דפי HTML.
- ניתן להוסיף קישור לספריית Bootstrap בתוך הפרויקט.
- נוסיף גם ספרייה `Img` לתמונות בתוך Pages.

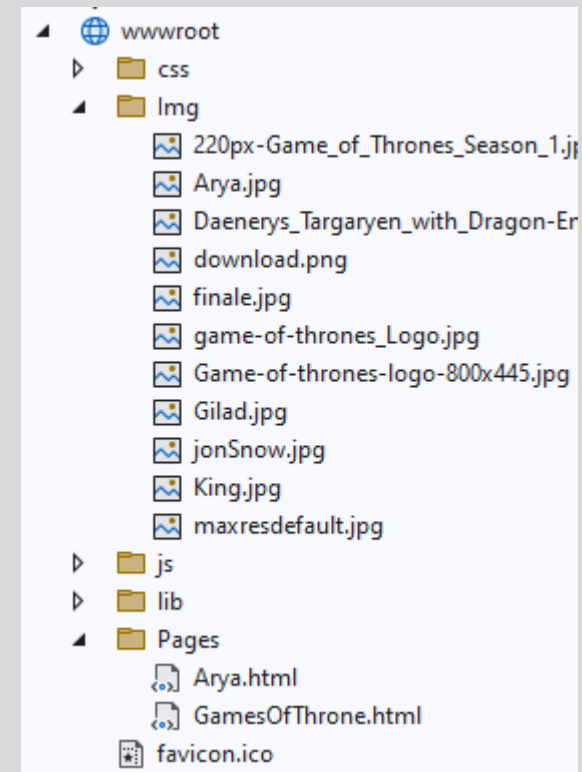
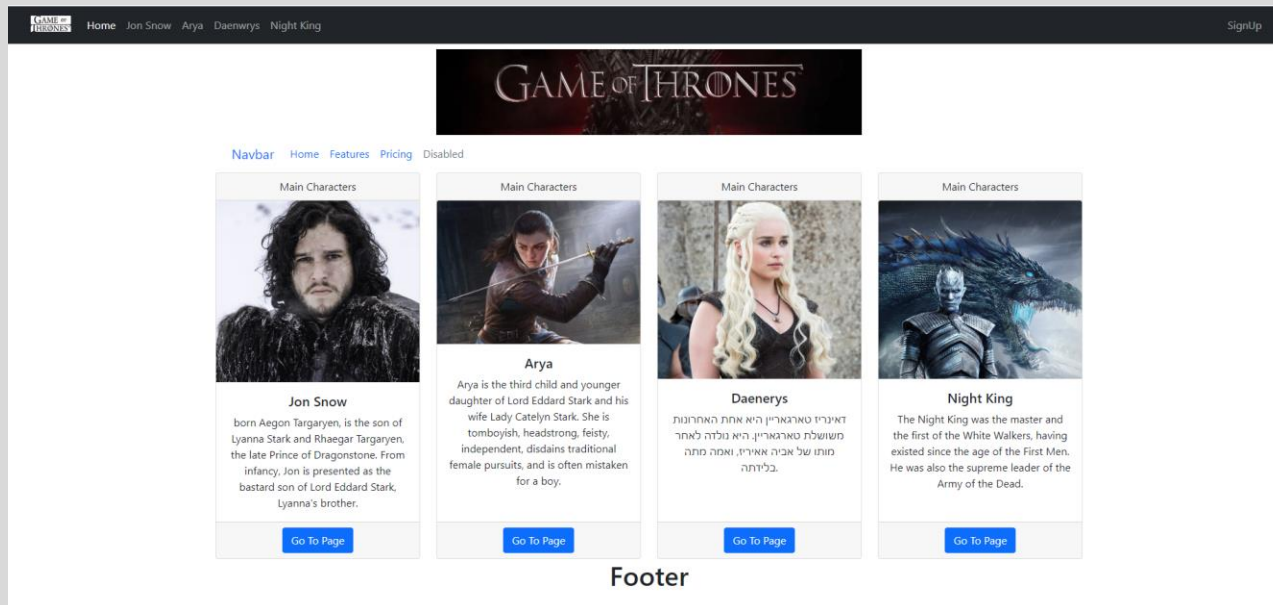


```

<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <title>Games Of Throne</title>
  <link href="../lib/bootstrap/dist/css/bootstrap.min.css" rel="stylesheet" />
</head>
<body>
  <div class="container text-center">...</div>
  <script src="../lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>

```

- הצגת דף HTML בפרויקט ASP.Net Core.
- ניתן להשתמש ב CSS וב JavaScript בדיוק כמו בכל דף HTML אחר, ובלבד שכל הקבצים ישמרו תחת ספרייה wwwroot.



שינוי הדף הראשי לדף סטטי

- ניתן להגדיר את דף HTML כדף הראשי של הפרויקט.
- נעשה זאת בדרך "עקומה קצת" אבל פשוטה. בקוד של דף Index נשנה את הפעולה OnGet כך:

```
public class IndexModel : PageModel
{
    private readonly ILogger<IndexModel> _logger;

    public IndexModel(ILogger<IndexModel> logger)
    {
        _logger = logger;
    }

    public IActionResult OnGet()
    {
        return Redirect("/Pages/GamesOfThrone.html");
    }
}
```



Razor Page

הוספת קוד C# ב- Razor view

- קוד צד שרת נכתב בדף באמצעות `@`.
- נוסף לדוגמה את התאריך והשעה הנוכחיים באמצעות קוד צד השרת לדף `.Index`.
- העברת נתונים מהקוד מאחורי לדף נעשית באמצעות הגדרת מאפיינים:
 - הגדרת מאפיינים במחלקה של הדף.
 - פניה למאפיין תעשה כך:

```
<div>@Model.Str</div>
```

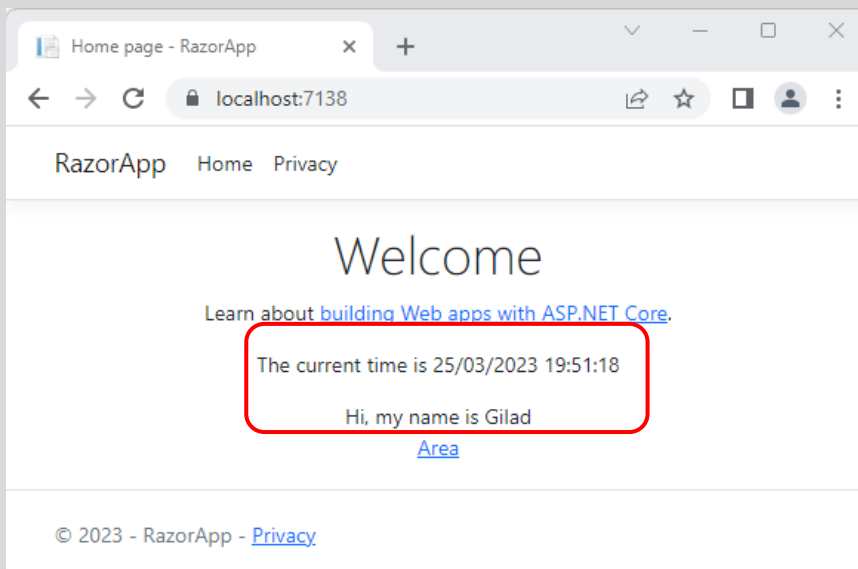
```
public class IndexModel : PageModel
{
    private readonly ILogger<IndexModel> _logger;

    public IndexModel(ILogger<IndexModel> logger)
    {
        _logger = logger;
    }

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

    public void OnGet() // IActionResult
    {
        //return Redirect("~/Pages/Area.html");
        Str = "Hi, my name is Gilad";
    }
}
```

```
<div class="text-center">
  <h1 class="display-4">Welcome</h1>
  <p>Learn about <a href="https://docs.microsoft.com/aspnet/core">ASP.NET Core</a>.</p>
  <p>The current time is @DateTime.Now</p>
  <div>@Model.Str</div>
  <a href="~/pages/area.html">Area</a>
</div>
```



Home page - RazorApp

localhost:7138

RazorApp Home Privacy

Welcome

Learn about [building Web apps with ASP.NET Core](#).

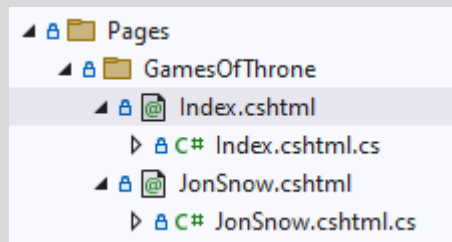
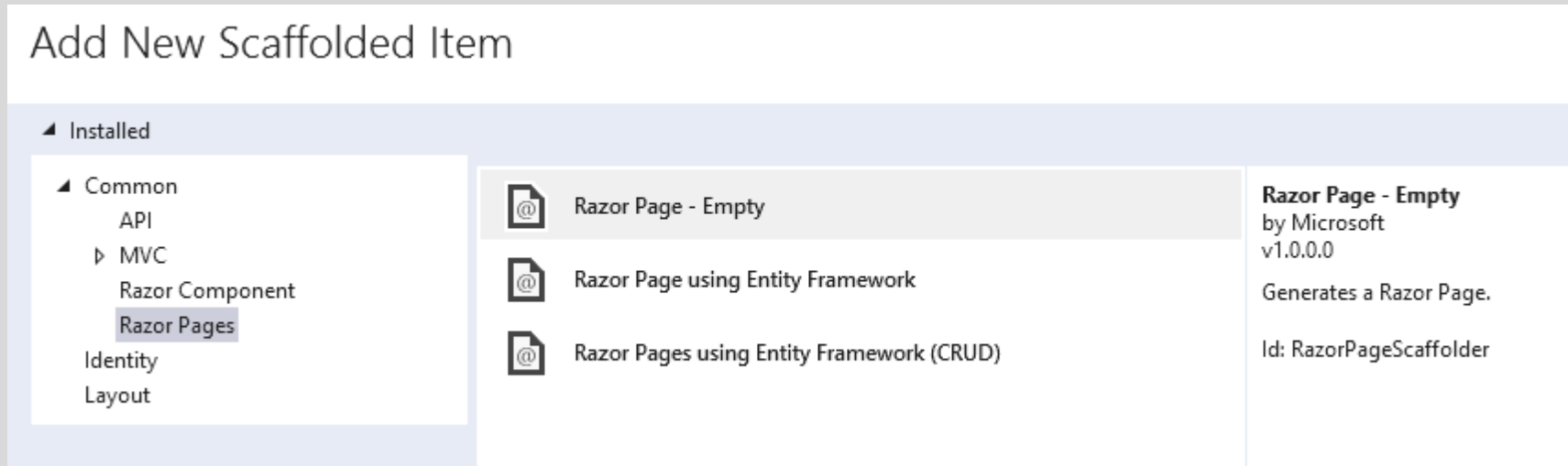
The current time is 25/03/2023 19:51:18

Hi, my name is Gilad

[Area](#)

© 2023 - RazorApp - [Privacy](#)

- נוסף ספריה בתוך Pages ונלחץ קליק ימני עליה ... Add-> Razor Page
- נבחר דף ריק ונציין את שם הדף עם הסיומת .cshtml



- נקבל דף עם קוד הצמוד אליו.

יצירת דף חדש – תבנית layout

- הכותרת של הדף היא מגדירה את הקשר בינו לבין הקוד הצמוד אליו.
- נריץ את הדף ונראה כי הוא מקושר לדף הראשי `_layout`.
- נשים לב שה `Title` לא תקיין. נוסיף:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device width, initial-scale=1.0" />
  <title>@ViewData["Title"] - RazorApp2</title>
  <link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />
  <link rel="stylesheet" href="~/css/site.css" asp-append-version="true" />
  <link rel="stylesheet" href="~/RazorApp2.styles.css" asp-append-version="true" />
</head>
<body>
  <header>...
  <div class="container">
    <main role="main" class="pb-3">
      @RenderBody()
    </main>
  </div>

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

  <script src="~/lib/jquery/dist/jquery.min.js"></script>
  <script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>
  <script src="~/js/site.js" asp-append-version="true"></script>

  @await RenderSectionAsync("Scripts", required: false)
</body>
</html>
```

```
@page
@model RazorApp.Pages.GamesOfThrone.TestModel
@{
  ViewData["Title"] = "Test";
}
```



```
@page
@model RazorApp.Pages.GamesOfThrone.IndexModel
@{
    ViewData["Title"] = "Index";
}

<div class="container text-center">
    <div class="row ">
        <nav class="navbar navbar-expand-md navbar-dark bg-dark fixed-top" data-bs-theme="dark">
            <div class="container-fluid">
                <a class="navbar-brand" href="/">...</a>
                <button class="navbar-toggler" type="button" data-bs-toggle="collapse">...</button>
                <div class="collapse navbar-collapse" id="navbarNav">
                    <ul class="navbar-nav">
                        <li class="nav-item">...</li>
                        <li class="nav-item">
                            <a class="nav-link" asp-page="/GamesOfThrone/JonSnow">Jon Snow</a>
                        </li>
                        <li class="nav-item">
                            <a class="nav-link" href="Pages/Area.html">Arya</a>
                        </li>
                        <li class="nav-item">...</li>
                        <li class="nav-item">...</li>
                    </ul>
                </div>
            </div>
        </nav>

    </div>
    <div class="row mt-5 pt-3">...</div>
    <div class="row">...</div>

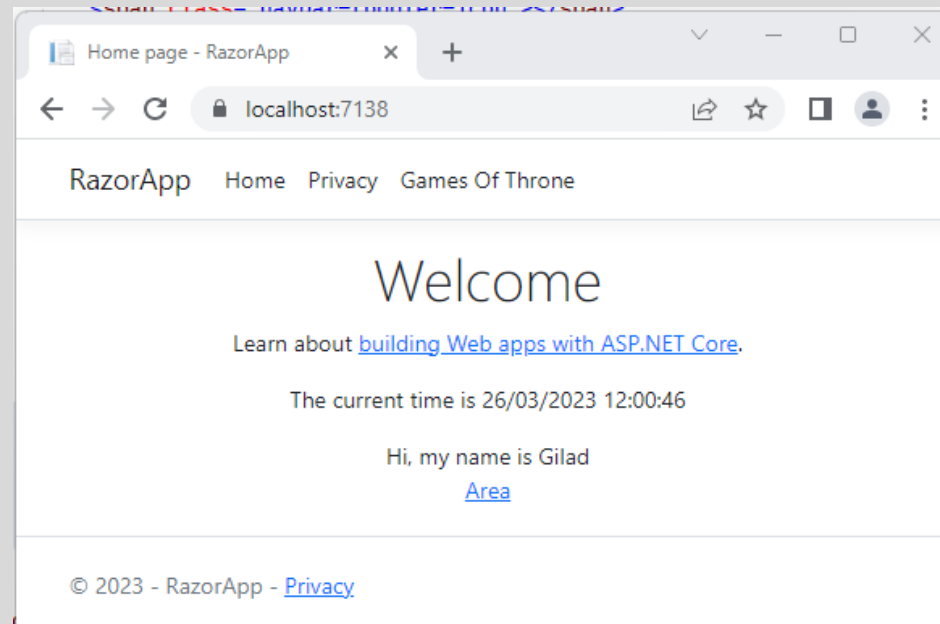
    <div class="row row-cols-1 row-cols-sm-2 row-cols-lg-3 row-cols-xl-4">...</div>

    <div class="row">...</div>
</div>
```

- נוסף את קוד HTML בשינויים הבאים:
- בותרת נוסף ל ViewData את שם הדף.
- קישור לדפי Razor אחרים יעשה באמצעות asp-page.
- התבנית של הדף הינה התבנית הראשית של הפרויקט Layout.
- התבנית הראשית כבר כוללת את ה Bootstrap כך שניתן להשתמש ללא קישור נוסף.

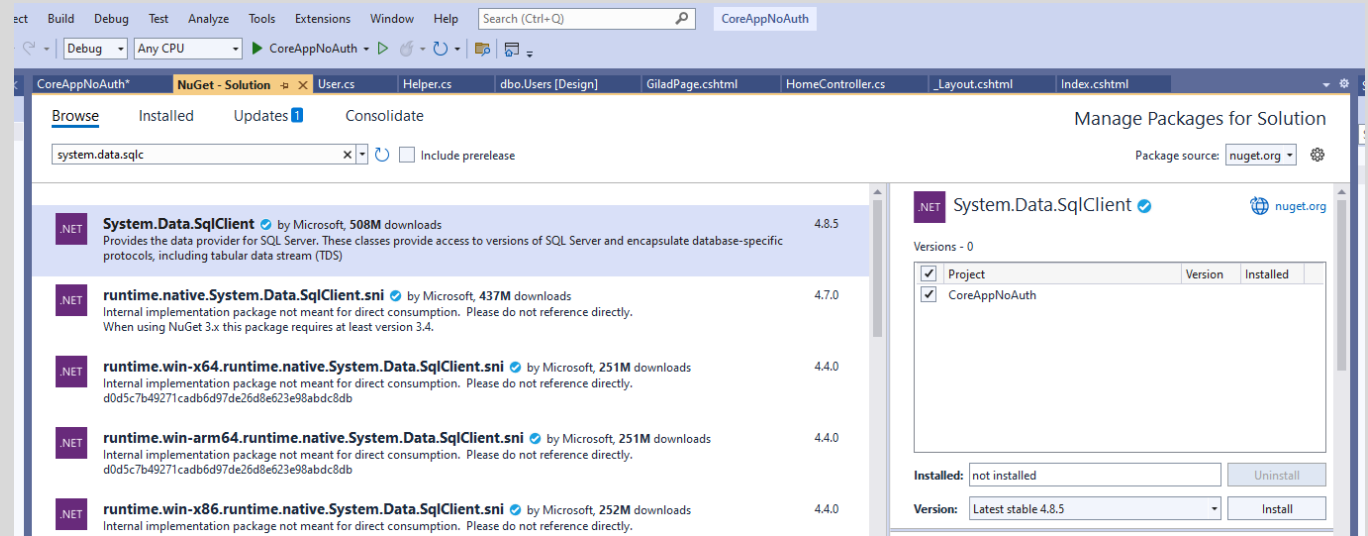
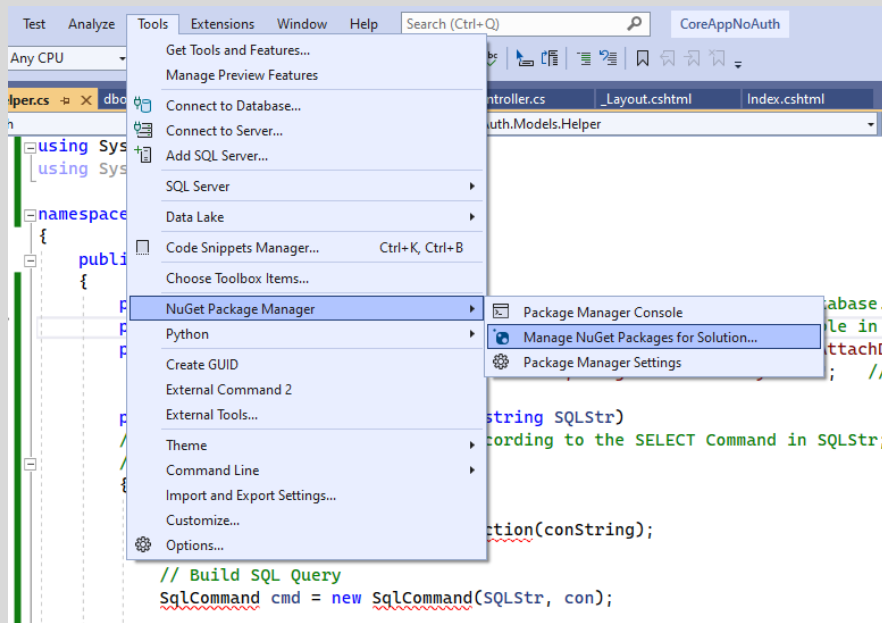
- נשנה את תבנית הראשית ונוסיף את הדפים החדשים לתפריט:

```
<li class="nav-item">  
  <a class="nav-link text-dark" asp-area="" asp-page="/Privacy">Privacy</a>  
</li>  
<li class="nav-item">  
  <a class="nav-link text-dark" asp-area="" asp-page="/GamesOfThrone/Index">Games Of Throne</a>  
</li>
```

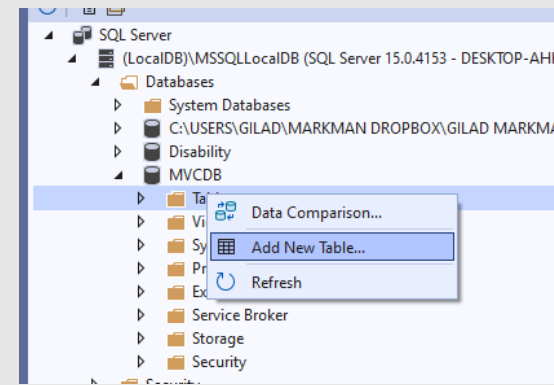
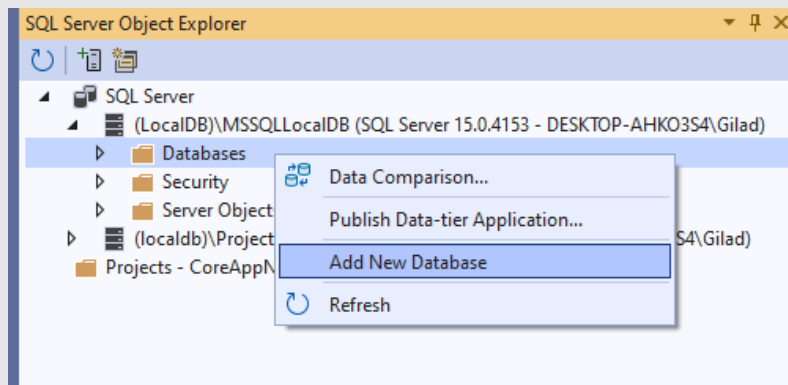


בסיס נתונים Ado.Net

- על מנת שנוכל להשתמש ב ADO.Net אנו צריכים להתקין חבילה מתאימה.
- נשתמש ב NuGet Package Manager. ונתקין את החבילה System.Data.SqlClient.



- נוסוף בסיס נתונים באמצעות SQL Server Object Explorer.
- קליק ימני על על databases ויצירת בסיס נתונים חדש.
- ולאחר מכן, קליק ימני על טבלאות ויצירת טבלה חדשה.
- ניצור טבלת משתמשים בדיוק כמו שלמדנו.



- ניצור ספריית Models ונוסיף שתי מחלקות:
- מחלקה של User התואמת את הטבלה שלנו.
- מחלקה של Helper בדומה למחלקה שהצגנו בקורס. נעדכן את הקוד בהתאם לטבלה שבנינו.

```
namespace CoreAppNoAuth.Models
{
    public class User
    {
        public int id { get; set; }
        public string Firstname { get; set; }
        public string Lastname { get; set; }
        public string Username { get; set; }
        public string Pass{ get; set; }
    }
}
```

- נעדכן את מחרוזת ההתקשרות ב Helper.
- במאפיינים של בסיס הנתונים נמצא את מחרוזת ההתחברות:

```
Data Source=(LocalDB)\MSSQLLocalDB;Initial Catalog=MVCDB;Integrated  
Security=True;Connect Timeout=30;Encrypt=False;Trust Server  
Certificate=False;Application Intent=ReadWrite;Multi Subnet Failover=False
```

- יש למחוק את המאפיינים המסומנים ולהעתיק לקוד של Helper במקום המתאים.

- נוסף ספריה UsersADO ובתוכה דף index מוגז Razor Page
- בקוד מאחורי יש שתי פונקציות עיקריות:
 - OnGet() – בכל פניה ראשונה לדף (לא postBack).
 - OnPost() – בכל פניה בעקבות postBack.
- העברת המידע לדף נעשית באמצעות מאפיינים.
- נוסף במחלקה מאפיין DataTable st
- נוסף את הקוד השולף את הטבלה ושומר במאפיין.

```
using Microsoft.AspNetCore.Mvc;  
using Microsoft.AspNetCore.Mvc.RazorPages;  
using RazorApp.Models;  
using System.Data;  
  
namespace RazorApp.Pages.UsersADO  
{  
    public class IndexModel : PageModel  
    {  
        public DataTable dt { get; set; }  
  
        public void OnGet()  
        {  
            string SQL = "SELECT * FROM Users";  
            DataSet ds = new DataSet();  
            ds = Helper.RetrieveTable(SQL);  
            dt = ds.Tables[0];  
        }  
    }  
}
```



```
@page
@model RazorApp.Pages.UsersADO.IndexModel
@using System.Data
@{ ViewData["Title"] = "UsersTable"; }
<h1>UsersTable</h1>
<table class='table'>
  <thead>
    <tr>
      @foreach (DataColumn column in Model.dt.Columns)
      {
        <th>@column.ColumnName</th>
      }
    </tr>
  </thead>
  <tbody>
    @foreach (DataRow row in Model.dt.Rows)
    {
      <tr>
        @foreach (DataColumn column in Model.dt.Columns)
        {
          <td>@row[column]</td>
        }
      </tr>
    }
  </tbody>
</table>
```

- בדף נוסף את האפשרות להשתמש באובייקט DataTable.

- נדפיס את הטבלה באמצעות קוד.

- כל שורה המתחילה ב @ מהווה קוד C# בצד השרת.

- דפי Razor יכולים לשלב קוד עם HTML.

- שימו לב שהשתמשנו בעיצוב Bootstrap.

[RazorApp](#) [Home](#) [Privacy](#) [Games Of Throne](#) [Users List](#) [Register](#)

UsersTable

| Id | Firstname | Lastname | Username | Pass |
|----|-----------|----------|----------|---------|
| 0 | Yoav | Markman | Y123 | Y123 |
| 1 | Gilad | Markman | Gilad | 1234 |
| 2 | Orly | Cohen | Orly | 1234 |
| 3 | Menny | Abudy | Menny | M123 |
| 4 | Lih | Markman | Lih123 | Lih123@ |

הוספת משתמשים

Register

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.RazorPages;
using RazorApp.Models;

namespace RazorApp.Pages.UsersADO
{
    public class RegisterModel : PageModel
    {
        public void OnGet()
        {
        }

        [BindProperty]
        public User User { get; set; } = default!;

        public IActionResult OnPost()
        {
            int n = Helper.Insert(User);
            return RedirectToPage("Index");
        }
    }
}
```

- נוסף דף Razor-Empty בשם Register.
- בקוד מאחור נוסף מאפיין User עם Data Annotation שבאמצעותו נעביר נתונים בין הדף לקוד.
- ה BindProperty מאפשר העברת נתונים מהדף לקוד מאחורי.
- בפונקציה onGet אנחנו לא עושים כלום. רק מציגים דף ריק.
- בפונקציה OnPost אנחנו מעדכנים את בסיס הנתונים, ולאחר מכן מעבירים לדף Index.

```
@page
@model RazorApp.Pages.UsersADO.RegisterModel
@{
    ViewData["Title"] = "Register";
}
<h1>Register</h1>

<h4>User</h4>
<hr />
<div class="row">
    <div class="col-md-4">
        <form method="post">
            <div asp-validation-summary="ModelOnly" class="text-danger"></div>
            <div class="form-group">
                <label asp-for="User.Firstname" class="control-label"></label>
                <input asp-for="User.Firstname" class="form-control" />
            </div>
            <div class="form-group">
                <label asp-for="User.Lastname" class="control-label"></label>
                <input asp-for="User.Lastname" class="form-control" />
            </div>
            <div class="form-group">
                <label asp-for="User.Username" class="control-label"></label>
                <input asp-for="User.Username" class="form-control" />
            </div>
            <div class="form-group">
                <label asp-for="User.Pass" class="control-label"></label>
                <input asp-for="User.Pass" class="form-control" />
            </div>
            <div class="form-group">
                <input type="submit" value="Create" class="btn btn-primary" />
            </div>
        </form>
    </div>
</div>

<div>
    <a asp-page="Index">Back to List</a>
</div>
```

• באמצעות ה `asp-for` אנחנו למעשה מקשרים בין הדף לבין הקוד. כל שינוי בדף יעדכן את המאפיין בקוד.

• התגיות הללו נקראות Tag Helpers.

• קיימים מספר תגיות לשימוש בדפי Razor.

Action Result



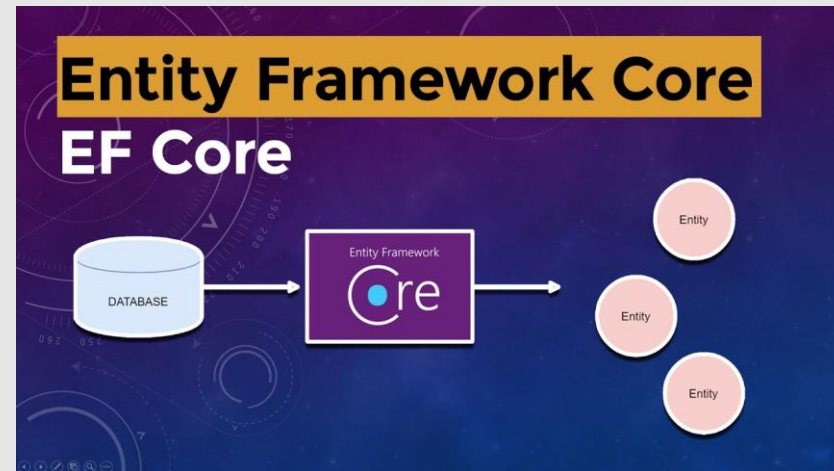
```
public IActionResult Index()
{
    //return Redirect("~/pages/testpage.html");
    return View();
}

public IActionResult Privacy()
{
    //return View("Privacy");
    return View();
}
```

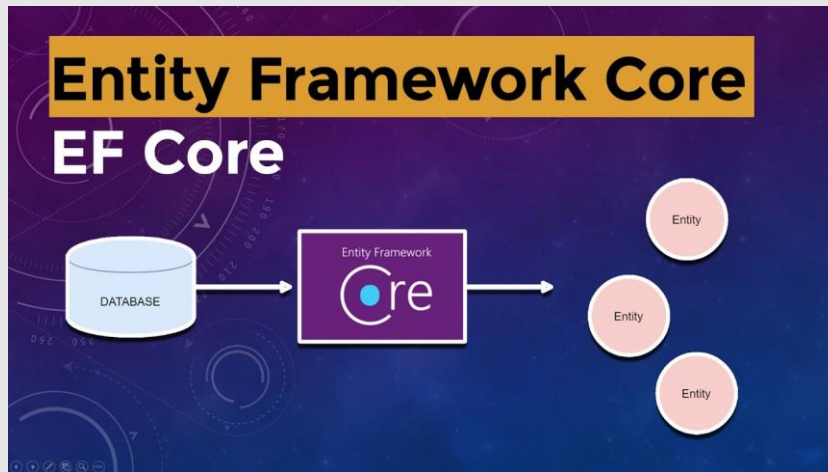
- ActionResult הוא ממשק הממומש
- על ידי מספר מחלקות:

| ActionResult | Helper Method | Description |
|-----------------------|-----------------|---|
| ViewResult | View | Renders a view as a web page |
| PartialViewResult | PartialView | Section of a view, that can be rendered inside another view |
| RedirectResult | Redirect | Redirect to another action method |
| RedirectToRouteResult | RedirectToRoute | Redirect to another action method |
| ContentResult | Content | Returns a user-defined content type |
| JsonResult | Json | Returns a serialized JSON object |
| JavaScriptResult | JavaScript | Returns a script that can be executed on the client |
| FileResult | File | Returns a binary output to write to the response |
| EmptyResult | (None) | returns a null result |

Entity Framework



- ORM-Object Relation Mapper של מיקרוסופט.
- Code First – הטכנולוגיה בונה את בסיס הנתונים בהתאם למחלקות (המודלים) שנבנים בקוד.
- Data First – הטכנולוגיה בונה את המחלקות (המודלים) בהתאם לבסיס נתונים קיים.



- נבנה מודל של Student – מחלקה המייצגת את הסטודנט.
- נוסף Data Annotation כדי להגדיר מאפיינים לבסיס הנתונים.
- חובה להוסיף שדה שימש כמפתח ראשי.

```
public class Student
{
    [Key]
    public int Id { get; set; }

    [Required]
    public string? Firstname { get; set; }

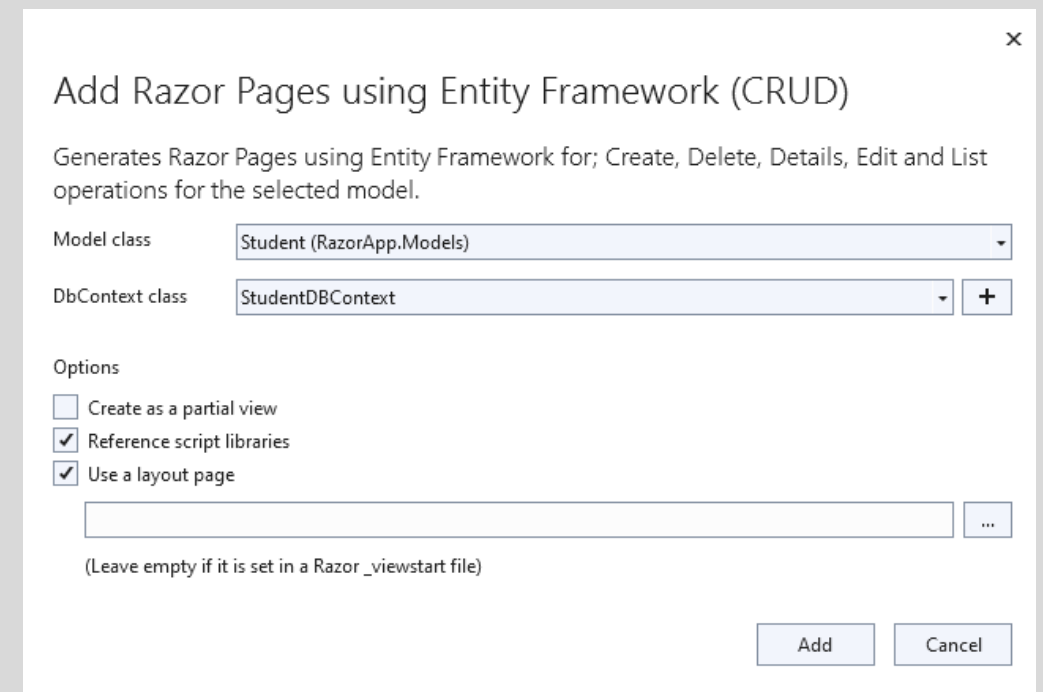
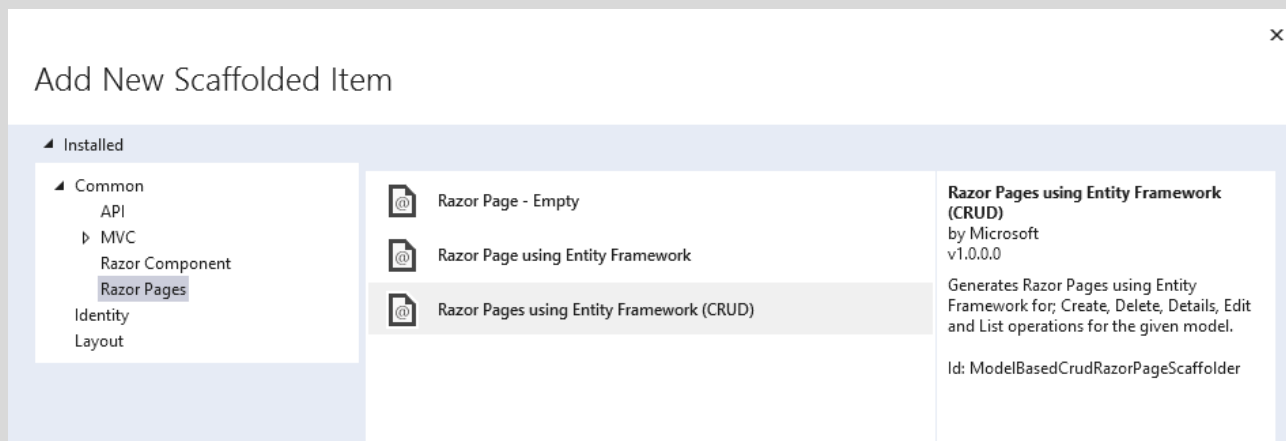
    [Required]
    public string? Lastname { get; set; }

    [Required]
    public string StudentClass { get; set; } = null!;

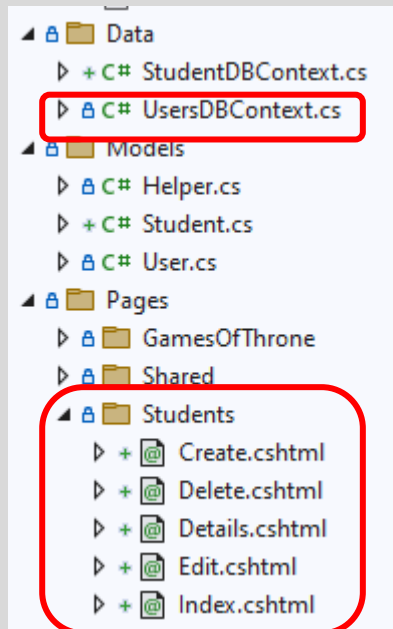
    [DataType(DataType.DateTime)]
    public DateTime Birthday { get; set; }

    public string Fullname()
    {
        return Lastname + " " + Firstname;
    }
}
```

- נפתח ב Pages ספריה חדשה. Students.
- קליק ימני על הספריה ונוסיף דף Razor באמצעות EF.
- נבחר את המודל Students וניצור DbContext חדש (באמצעות +).



יצירת התצוגות וה Context



- לחיצה על Add והמערכת תבנה עבורנו מספר קבצים.
- קבצי Razor עבור הפעולות הבסיסיות מול בסיס נתונים.
- קובץ Context שהוא מהווה קובץ עם הפעולות לעבודה מול בסיסי הנתונים, בהתאם לטכנולוגיה של EF.
- בקובץ Program יתווסף שירות המגדיר את בסיס הנתונים.

```
builder.Services.AddDbContext<StudentDBContext>(options =>  
options.UseSqlServer(builder.Configuration.GetConnectionString("StudentDBContext") ?? throw new InvalidOperationException("Connection string 'StudentDBContext' not found.")));
```

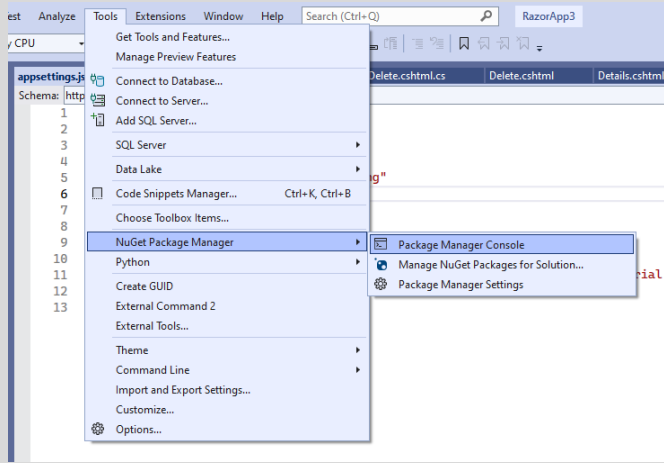
- בקובץ ה appsettings.json הוספה הגדרה למחרוזת התחברות ללא שם בסיס הנתונים:

```
"ConnectionStrings": {  
  "StudentsDBContext": "Server=(localdb)\\mssqllocaldb;Database=;Trusted_Connection=True;MultipleActiveResultSets=true"  
}
```

- ניצור בסיס נתונים חדש באמצעות קליק ימני ב SQL Server Object Explorer או שנחבר לבסיס נתונים קיים על ידי כך שנוסיף את שם בסיס הנתונים.

```
"ConnectionStrings": {  
  "StudentsDBContext": "Server=(localdb)\\mssqllocaldb;Database=EFTutorial;Trusted_Connection=True;MultipleActiveResultSets=true"  
}
```

- נבצע migration כדי לעדכן באמצעות Entity Framework את הקוד ואת בסיס הנתונים.



- נפתח את Package Manager Console.

- נבצע את שתי הפקודות הבאות:

- Add-migration "migration_name" –Context StudentsDBContext
- Update-database –Context StudentsDBContext

הרצת הפרויקט

- נריץ הפרויקט ונכנס לדף localhost:xxx/Students

Index

[Create New](#)

| Firstname | Lastname | StudentClass | Birthday |
|-----------|----------|--------------|----------|
|-----------|----------|--------------|----------|

Index

[Create New](#)

| Firstname | Lastname | StudentClass | Birthday | |
|-----------|----------|--------------|--------------------|---|
| גלעד | מרקמן | יא | 12/12/2020 0:00:00 | Edit Details Delete |

- ניצור סטודנט חדש בדאטה בייס:

• יצרנו בכמה לחיצות פרויקט הכולל בסיס נתונים, וכל הדפים הנדרשים לנהל את בסיס הנתונים:

• רשימת הנתונים.

• הוספת נתונים.

• מחיקת נתונים.

• עדכון נתונים.

• מסך לקבלת פרטים מלאים.

Details

Student

| | |
|--------------|--------------------|
| Firstname | גלעד |
| Lastname | מרקמן |
| StudentClass | יא |
| Birthday | 12/12/2020 0:00:00 |

[Edit](#) | [Back to List](#)

Create

Student

Firstname

Lastname

StudentClass

Birthday

Create

[Back to List](#)

אם יש זמן נעבור ל MVC