תכנות באינטרנט

Backend

Razor pages

גלעד מרקמן

קריית החינוך פארק המדע,
נס ציונה

Microsoft
ASP.NET Core

GM
Internet Programing
Courses

כתיבת קוד צד שרת
C#

הוספת קוד #C ב- Razor view

```
@page
@model RazorApp2.Pages.RazorPages.Razor1Model
@{
}
<div>
    <h2>The current time is @DateTime.Now</h2>
</div>
<div>
    @{
        for (int i = 0; i < 10; i++)
        {
            if (i % 2 == 0)
            {
                <div style="color:red">@i</div>
            }
            else
            {
                <div style="color:blue">@(i * 2)</div>
            }
        }
        <div style="background-color:aquamarine">The end</div>
        @Html.Raw("This is a eay to write tags and text")
    }
</div>
```

- קוד צד שרת יכול להכתב שני מקומות:
  - בדף Razor View באמצעות @.
  - בדף cs. הצמוד – code behind.
- באמצעות @ ניתן לכתוב כל קוד של #C ולשלב בתוכו הדפסה של תגיות HTML.
- דוגמה – הוספת תאריך ושעה נוכחיים.
- דוגמה – שימוש בלולאה המדפיסה תגיות HTML מעוצבות.

The current time is 15/06/2024 12:50:36

0
2
2
6
4
10
6
14
8
18

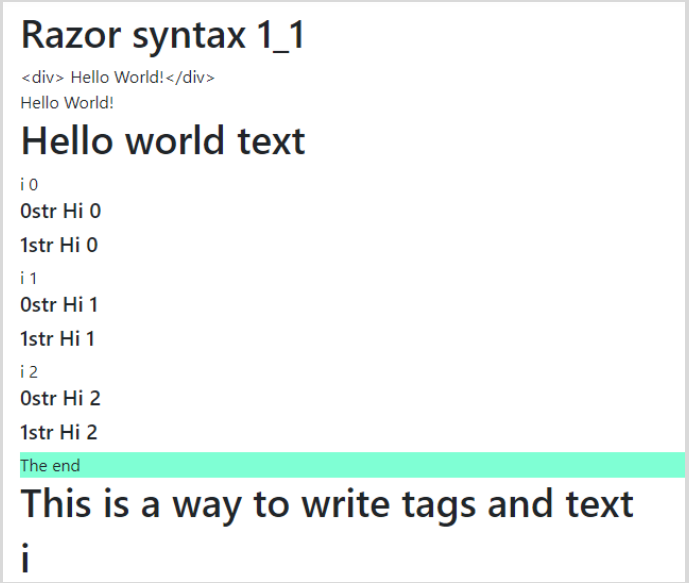
The end

This is a eay to write tags and text

Razor syntax

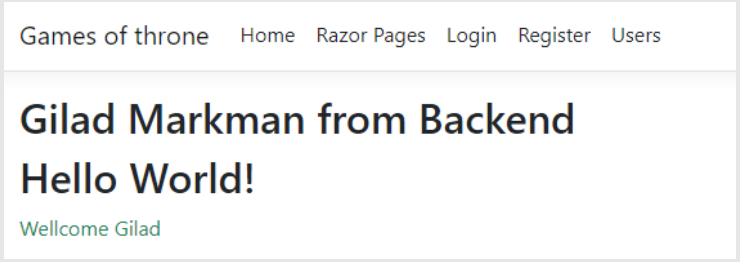
```
<h1>Razor syntax 1_1</h1>
@{
    string tag = "<div> Hello World!</div>";
    string txt = "Hello world text";
}
@tag
@Html.Raw(tag)
<h1>@txt</h1>
<div>
    @{
        for (int i = 0; i < 3; i++)
        {
            <div>
                i @i
                @{
                    string str = "Hi " + i;
                    for(int j=0; j < 2; j++)
                    {
                        <h5><span>@j</span><span>str @str</span></h5>
                    }
                }
            </div>
        }
        <div style="background-color:aquamarine">The end</div>
        @Html.Raw("<h1>This is a way to write tags and text</h1>")
    }
@{
    <h1>i</h1>
}
```

- @var – מדפיס את ערך המשתנה לדף.
- @{} – פותח בלוק של קוד בסי שארפ
  - בתוך תגית יש להתחיל מחדש בלוק
- @Html.Raw(str) – מדפיס לדף תגיות.
- @ { <tag></tag> } – כתיבת תגית בתוך בלוק מדפיסה את התגית בדף.



העברת נתונים מצד השרת -> לדף

- העברת נתונים מצד השרת לדף נעשית באמצעות הגדרת מאפיינים, ופניה למאפיין באמצעות האובייקט Model.
- בדף ניתן להשתמש גם בפעולות המוגדרות במודל.




```cshtml
@page
@model RazorApp2.Pages.RazorPages.Razor2Model
@{
}
<h2>@Model.Str</h2>
<h2>@Model.printText()</h2>
@if (Model.User == "Gilad")
{
    <div class="text-success">Wellcome @Model.User</div>
}
else
{
    <div class="text-danger">Go away</div>
}
```

```cshtml.cs
public class Razor2Model : PageModel
{
    public string Str { get; set; } = string.Empty;
    public string User { get; set; } = string.Empty;

    public void OnGet()
    {
        Str = "Gilad Markman from Backend";
        User = "Gilad";
    }
    public string printText()
    {
        return "Hello World!";
    }
}
```

ViewData

- דרך נוספת להעביר נתונים לדף היא באמצעות אובייקט ViewData.
- ViewData הוא מילון לו ניתן להוסיף מפתחות וערכים כראות עינינו באמצעות הפעולה:

```
ViewData["Msg"] = "This is a message";
ViewData["Number"] = 5;
```

- הערכים במילון הם Objects ולכן ניתן לשמור בו כל ערך. השימוש בערך מחייב המרה (המרה למחרוזת נעשית אוטומטית עם @).

```
public class Razor3Model : PageModel
{
    public void OnGet()
    {
        ViewData["Msg"] = "This is a message";
        ViewData["Number"] = 5;
    }
}
```

```
@page
@model RazorApp2.Pages.RazorPages.Razor3Model
@{
    ViewData["Title"] = "Razor3";
}
<h1>@ViewData["Title"]</h1>
<h3 class="alert-primary">@ViewData["Msg"]</h3>
<h3>@((int)ViewData["Number"]+1)</h3>
```

ViewData

- נשים לב שה Layout עושה שימוש ב ViewData["Title"] המוגדר בדף עצמו לקביעת הכותרת.
- האובייקט ViewData מתאפס בכל קריאה לדף – בכל Request.
- לכן, לא ניתן להשתמש בו להעברת מידע בין דפים.


```
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ViewData["Title"] - Games of throne</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>
```

הפעולות OnGet, OnPost

- הפעולות OnGet, OnPost מנהלות את פעולות ה Request.
- קריאה לדף באמצעות קישור - מבצעת פעולת Get ולכן תטופל על ידי OnGet. על כן, כניסה ראשונית לדף תגיע לפעולה זו.
- קריאה לדף באמצעות Form תבצע פעולה כפי שמוגדרת במאפיין method בטופס. בד"כ נשתמש ב post.

```
public class Razor_OnPostModel : PageModel
{
    public void OnGet()
    {
    }
    public void OnPost()
    {
    }
}
```

```
<form method="post">
    <input type="text" name="txt" />
    <input type="submit" value="submit"/>

</form>
```

OnPost & Action

- Asp.Net מחייב כי בכל קריאת Post יצורף לטופס AntiForgeryToken.
- Cross-Site Request Forgery
  - Razor Pages מוסיף את ה Token בצורה אוטמטית בקריאת Post לאותו דף.
  - אם מבצעים קריאת Post לדף אחר באמצעות action יש להוסיף את תגית האבטחה בעצמנו אחרת מקבלים שגיאת Http Error 400.

```
<form method="post" action="Razor_OnPost/">
    @Html.AntiForgeryToken()
    <input type="text" name="txt" />
    <input type="submit" value="submit" />

</form>
```

```
<form method="post">
  <input type="text" name="txt"> == $0
  <input type="submit" value="submit">
  <input name="__RequestVerificationToken" type="hidden" value="CfDJ8CFivx28y_FEuCladJr
  JPOcoOmKS64-K7DDSeYf_OiByIm0XDWGtGOiuJxeb8D10Bo-SOdYT2ZhiVhakQBpofyGpF7Oeljxb4UNXU2s5
  018sSTfSKtt2tzeDaGieeGN8RQZmh0YBlc9ZlWzIOUwrhZo">
</form>
```

Redirect

- הפעולות OnGet, OnPost מחזירות את התשובה למשתמש.
- אם הפעולה אינה מחזירה ערך (void) המערכת תחזיר את ה View של הדף כקובץ HTML.
- אם רוצים להפנות את המשתמש לדף אחר יש להחזיר אובייקט מתאים כדלקמן:

```
public void OnGet() {

}
public IActionResult OnPost()
{
    return RedirectToPage("/AccessDenied");
}
```

Stateless Programing

Session Object

תכנות חסר מצב

- השרת אינו שומר נתונים על ההתקשרות עם הלקוח.
- לאחר שהלקוח שלח Request וקיבל Response הקשר עם המשתמש (הדפדפן) "מנותק" והשרת אינו שומר נתונים על הדף שנשלח.
- שיטה זו מאפשר לשרת לטפל במקביל במספר רב של משתמשים.

אובייקט Session

- לתכנות חסר מצב יש מספר חסרונות:
  - לא נשמרים נתונים לגבי המשתמש – כיצד לדוגמה נבדוק אם המשתמש ביצע כניסה (Login) בקריאות הקודמות שלו.
  - לא ניתן להעביר נתונים מדף אחד לדף שני – כל דף מהווה קריאה נפרדת וכל המידע נמחק.

- הפתרון – אובייקט מיוחד גלובלי הנקרא Session.

אובייקט Session

- Session (שיחה) – מוגדר כפרק הזמן שבו המשתמש פונה לאתר ועד אשר הוא מתנתק ממנו.
- אובייקט Session הוא מילון בו ניתן לשמור נתונים של כל משתמש בנפרד.
- האובייקט קיים במהלך זמן השיחה של המשתמש.
- אובייקט זה הוא אישי למשתמש מסוים, בשיחה מסוימת. לכל משתמש יש אובייקט Session משלו. אך האובייקט גלובלי ומשותף לכל דפי האתר.
- סיום ה session מתרחש לאחר פרק זמן שנקבע מראש בו המשתמש לא ביצע פניה (Request) לשרת.
- פרטי ה Session נשמרים על הדפדפן באמצעות Cookie, וכך השרת יודע מי המשתמש שפונה אליו.

הגדרת Session ב Asp.Net Core

- כדי להשתמש באובייקט Session עלינו להגדיר אותו תחילה בקובץ הראשי Program.cs:

```
// Add Session service
builder.Services.AddDistributedMemoryCache();
builder.Services.AddSession(options =>
{
    options.IdleTimeout = TimeSpan.FromMinutes(10);
    options.Cookie.HttpOnly = true;
    options.Cookie.IsEssential = true;
});
```

- זמן לסיום נקבע ל- 10 דקות.

```
// Add Session service
app.UseSession();

app.MapRazorPages();
```

- הסבר לשימוש ב Session

שימוש ב Session – דפי View

- אובייקט Session הוא מילון השומר מחרוזות ושלמים בלבד.
- ניתן כל העת להוסיף ערכים חדשים או לשנות ערכים קיימים.
- בדפי Razor View אנחנו נשתמש באובייקט Context.
- דוגמה: דף _Layout.

```
@using Microsoft.AspNetCore.Http
@{
    var sessionValue = Context.Session.GetString("Login");
}
```

```
@if (string.IsNullOrEmpty(sessionValue))
{
    <span class="navbar-text">wellcome Visitor</span>
}
else
{
    <span class="navbar-text">wellcome @sessionValue</span>
}
```

שימוש ב Session בקוד

- בקוד cs אנחנו נעשה שימוש באובייקט HttpContext אשר כולל את הפעולות הבאות:

```
var value = HttpContext.Session.Get("Login");
int? num = HttpContext.Session.GetInt32("Login");
string? str = HttpContext.Session.GetString("Login");
HttpContext.Session.SetInt32("Id", 25);
HttpContext.Session.SetString("Login", userName);
```

- הוספת ? (int?) יוצרת משתנה שיכול לקבל גם ערך Null.

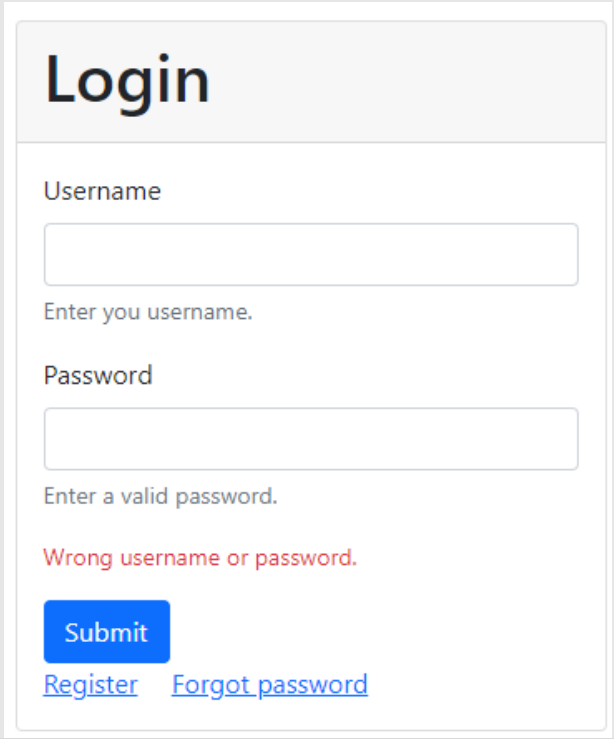
קליטת מידע מצד המשתמש
לצד השרת

שימוש בטפסים

קליטת מידע מהמשתמש

- ניתן לקלוט מידע מטופס (Form) מצד המשתמש לצד השרת בשלוש דרכים שונות:
- א. שימוש באובייקט Request.Form – שיטה ישנה ולא מומלצת
- ב. שימוש בפרמטרים בפעולות OnPost, OnGet.
- ג. קישור מאפיינים במודל לדף [BindProperty].

- נבנה תחילה טופס כניסה להדגמה.



דף כניסה – צד המשתמש

```
@page
@model RazorApp2.Pages.Users.LoginModel
@{
    ViewData["Title"] = "Login";
}
<div class="row justify-content-center align-items-center" style="height:75vh">
    <div class="col-sm-10 col-md-8 col-lg-6 col-xl-4">
        <div class="card h-100">
            <div class="card-header"><h1>Login</h1></div>
            <div class="card-body">
                <form method="post">
                    <div class="mb-3">
                        <label for="Username" class="form-label">Username</label>
                        <input type="text" class="form-control" id="Username" name="Username">
                        <div id="UsernameHelp" class="form-text">Enter you username.</div>
                    </div>
                    <div class="mb-3">
                        <label for="password" class="form-label">Password</label>
                        <input type="password" class="form-control ms-auto" id="password" name="password">
                        <div id="PasswordHelp" class="form-text">Enter a valid password.</div>
                    </div>
                    <div class="mb-3 form-text text-danger">
                        @Model.msg
                    </div>

                    <button type="submit" class="btn btn-primary">Submit</button>
                </form>
                <a href="#" class="card-link">Register</a>
                <a href="#" class="card-link">Forgot password</a>
            </div>
        </div>
    </div>
</div>
```

- נכין דף כניסה עם טופס מתאים.
- נוסיף מקום להודעות מהשרת.
- מומלץ לממש גם את צד המשתמש JS, ולעשות בדיקה לטופס לפני שליחתו לשרת.

שימוש ב Request.Form

- האובייקט Request.Form הינו אובייקט שנשלח על ידי הטופס לשרת בעת ביצעו פעולה Post.
- האובייקט עוטף את כל הנתונים שהוכנסו לטופס, ובצד השרת ניתן לראות את הערכים בטופס.
- שיטה זו אינה מומלצת עוד.

```
public IActionResult OnPost()
{

    if (Request.Form["Username"] == "Gilad" && Request.Form["password"] == "1968")
    {
        HttpContext.Session.SetString("Login", Request.Form["Username"]);
        HttpContext.Session.SetString("Admin", "True");
        return RedirectToPage("/Index");
    }
    msg = "Wrong username or password.";
    return Page();
}
```

שימוש בפרמטרים בפעולות

```
public class Login2Model : PageModel
{
    public string msg { get; set; } = string.Empty;
    public void OnGet()
    {
    }
    public IActionResult OnPost(string Username, string password)
    {
        if (Username == "Gilad" && password == "1968")
        {
            HttpContext.Session.SetString("Login", Username);
            HttpContext.Session.SetString("Admin", "True");
            return RedirectToPage("/Index");
        }
        msg = "Wrong username or password.";
        return Page();
    }
}
```

```
<input type="text" class="form-control" id="Username" name="Username">
```

- הערכים של השדות בטופס מועברים כפרמטרים לפונקציה OnPost.
- הקישור Binding נעשה באמצעות המאפיין name של השדה בטופס.

שימוש במאפיינים של המודל

```cs
public class Login3Model : PageModel
{
    public string msg { get; set; } = string.Empty;

    [BindProperty]
    public string Username { get; set; }

    [BindProperty]
    public string password { get; set; }

    public void OnGet()
    {
    }
    public IActionResult OnPost()
    {
        if (Username == "Gilad" && password == "1968")
        {
            HttpContext.Session.SetString("Login", Username);
            HttpContext.Session.SetString("Admin", "True");
            return RedirectToPage("/Index");
        }
        msg = "Wrong username or password.";
        return Page();
    }
}
```

- הגדרה במודל של מאפיינים בשם זהה ל name של השדה בטופס.
- בנוסף, יש להוסיף לפני כל מאפיין [BindProperty].
- בדרך זו הקישור הוא דו כווני. כל עדכון בטופס מופיע במאפיין ולהפך.
- יש להוסיף לתגית את המאפיין value כדי שיציג את הנתון.

```cs
<input type="text" class="form-control" id="Username" name="Username" value="@Model.Username">
```

Tag Helpers

- ה Tag Helpers הם פקודות שנועדו לעזור לנו לכתוב את קוד HTML.
- בעת יצירת הדף הפקודות מוחלפות במאפיינים הנדרשים ב HTML, וכך חוסכים בכתיבה ומונעים טעויות.
- אין חובה להשתמש ב Tag Helpers, וניתן לערוך את הטופס עם HTML.

```
<form method="post">
    <div class="mb-3">
        <label asp-for="Username" class="form-label">Username</label>
        <input asp-for="Username" class="form-control">
        <div id="UsernameHelp" class="form-text">Enter you username.</div>
    </div>
    <div class="mb-3">
        <label asp-for="password" class="form-label">Password</label>
        <input asp-for="password" class="form-control ms-auto">
        <div id="PasswordHelp" class="form-text">Enter a valid password.</div>
    </div>
    <div class="mb-3 form-text text-danger">
        @Model.msg
    </div>
    <button type="submit" class="btn btn-primary">Submit</button>
</form>
```

```
<label class="form-label" for="Username">Username</label>
<input class="form-control" type="text" data-val="true" data-val-required=
"The Username field is required." id="Username" name="Username" value>
<div id="UsernameHelp" class="form-text">Enter you username.</div>
```

דוגמה

הרשאות כניסה
ואבטחת דפים

הפרויקט ב GitHub

הדגמת - מערכת הרשאות ואבטחת דפים

- שם המשתמש והסיסמה בשלב זה יכתבו בקוד. בפרקים הבאים נשתמש בבסיס נתונים.
- המערכת תכלול:
  - משתנה Session[“Login”] אשר יכלול את שם המשתמש שביצע כניסה.
  - משתנה Session[“Admin”] אשר יקבל ”True“ אם המשתמש מנהל מערכת.
  - טופס כניסה שיבדוק שם משתמש וסיסמה ויעדכן את אובייקט ה session.
  - בתבנית הראשית ייכתב שם המשתמש שביצע כניסה או ”Visitor“.
  - יקבעו דפים שהכניסה תותר רק למשתמש רשום או למנהל מערכת.
  - יתווסף כפתור יציאה – Logout.

התקנת אובייקט Session

```
var builder = WebApplication.CreateBuilder(args);

// Set the DataDirectory to the App_Data folder
var dataDirectory = Path.Combine(builder.Environment.ContentRootPath, "App_Data");
AppDomain.CurrentDomain.SetData("DataDirectory", dataDirectory);

// Add services to the container.
builder.Services.AddRazorPages();

// Add Session service
builder.Services.AddDistributedMemoryCache();
builder.Services.AddSession(options =>
{
    options.IdleTimeout = TimeSpan.FromMinutes(10);
    options.Cookie.HttpOnly = true;
    options.Cookie.IsEssential = true;
});
var app = builder.Build();

// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Error");
    // The default HSTS value is 30 days. You may want to change this for productior
    app.UseHsts();
}

app.UseHttpsRedirection();
app.UseStaticFiles();
app.UseRouting();
app.UseAuthorization();

// Add Session service
app.UseSession();

app.MapRazorPages();
app.Run();
```

- בתוכנית Program.cs יש להוסיף את הקוד של שימוש ב Session.
- לשים לב להקליד את הפקודות במקום הנכון בקוד.
- GitHub - Program.cs

דף כניסה – צד המשתמש

```razor
@page
@model RazorApp2.Pages.Users.Login4Model
@{
    ViewData["Title"] = "Login4";
}


<div class="row justify-content-center align-items-center" style="height:75vh">
    <div class="col-sm-10 col-md-8 col-lg-6 col-xl-4">
        <div class="card h-100">
            <div class="card-header"><h1>Login 4</h1></div>
            <div class="card-body">
                <form method="post">
                    <div class="mb-3">
                        <label asp-for="Username" class="form-label">Username</label>
                        <input asp-for="Username" class="form-control">
                        <div id="UsernameHelp" class="form-text">Enter you username.</div>
                    </div>
                    <div class="mb-3">
                        <label asp-for="password" class="form-label">Password</label>
                        <input asp-for="password" class="form-control ms-auto">
                        <div id="PasswordHelp" class="form-text">Enter a valid password.</div>
                    </div>
                    <div class="mb-3 form-text text-danger">
                        @Model.msg
                    </div>
                    <button type="submit" class="btn btn-primary">Submit</button>
                </form>
                <a href="#" class="card-link">Register</a>
                <a href="#" class="card-link">Forgot password</a>
            </div>
        </div>
    </div>
</div>
```

- נכין דף כניסה עם טופס מתאים.
- נוסיף מקום להודעות מהשרת.
- מומלץ לממש גם את צד המשתמש JS, ולעשות בדיקה לטופס לפני שליחתנו לשרת.

דף כניסה – צד השרת – Request.Form

```
public class Login4Model : PageModel
{
    public string msg { get; set; } = string.Empty;
    [BindProperty]
    public string Username { get; set; }
    [BindProperty]
    public string password { get; set; }

    public void OnGet()
    {
    }
    public IActionResult OnPost()
    {
        if (Username == "Gilad" && password == "1968")
        {
            HttpContext.Session.SetString("Login", Username);
            HttpContext.Session.SetString("Admin", "True");
            return RedirectToPage("/Index");
        }
        msg = "Wrong username or password.";
        return Page();
    }
}
```

- נוסיף מאפיינים שיקושרו לשדות בטופס Username, password.
- נוסיף מאפיין (חד כיווני) של הודעה במקרה של כניסה שגויה.
- בפעולה post לאחר לחיצה על submit, אם שם המשתמש והסיסמה נכונים נעדכן את ה Session בהתאם, ונעבור לדף הראשי.
- אם הכניסה נכשלה נוסיף הודעת שגיאה ונחזיר את הדף הנוכחי.

תבנית ראשית - _Layout

- בראש הדף בקוד CSharp נגדיר משתנה עם שם המשתמש.

```
@using Microsoft.AspNetCore.Http
@{
    var sessionValue = Context.Session.GetString("Login");
}
```

- בקצה הימני של התפריט נוסיף הדפסה Welcome Username.

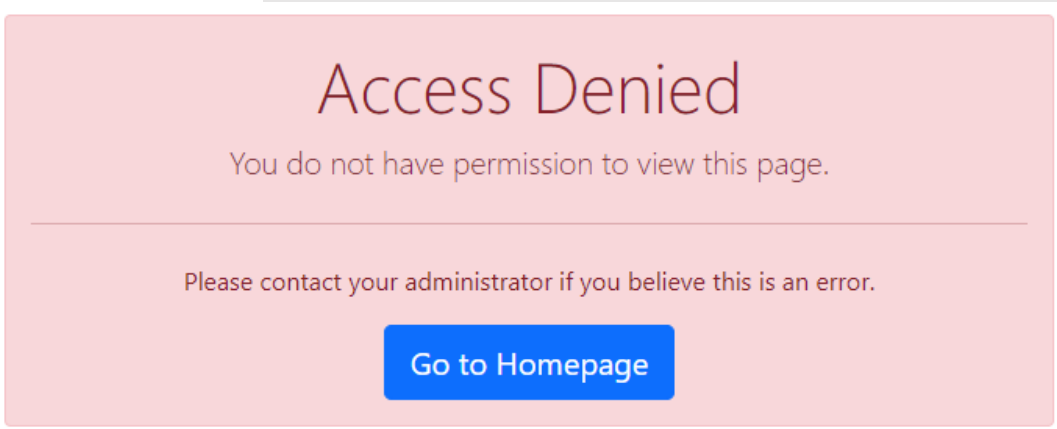
```
@if (string.IsNullOrEmpty(sessionValue))
{
    <span class="navbar-text">wellcome Visitor</span>
}
else
{
    <span class="navbar-text">wellcome @sessionValue</span>
}
```

- נוסיף לתפריט קישור לדף הכניסה. קובץ התבנית ב GitHub

אבטחת דפים – משתמש רגיל

- נוסיף בדפים אותם אנחנו מעוניינים לקבוע הרשאות את הפקודה אשר מעבירה את המשתמש שאינו בעל הרשאה לדף מיוחד.

```
public class DaenerysModel : PageModel
{
    public IActionResult OnGet()
    {
        if (string.IsNullOrEmpty(HttpContext.Session.GetString("Login")))
        {
            return RedirectToPage("/AccessDenied");
        }
        return Page();
    }
}
```



אבטחת דפים – מנהל מערכת

- בדפים בהם אנחנו רוצים לקבוע הרשאה מיוחדת, למשל רק מנהל מערכת, נבצע בדיקה מול אובייקט ה Session המתאים.
- תשומת לב – אובייקט Session מקבל רק מחרוזות ומספרים שלמים. לכן שמרנו בדף הכניסה מחרוזת של "True" אם המדובר במנהל מערכת. אחרת המפתח לא קיים (מחזיר null).

```
public IActionResult OnGet()
{
    if (HttpContext.Session.GetString("Admin") != "True")
    {
        return RedirectToPage("/AccessDenied");
    }
```

Logout

- יציאה של משתמש Logout נעשתה בדרך הבאה:
  - הוסף קישור (בצורת כפתור) בתפריט הראשי המוצג רק אם בוצעה כניסה.

```
@if (string.IsNullOrEmpty(sessionValue))
{
    <span class="navbar-text">welcome Visitor</span>
}
else
{
    <span class="navbar-text">welcome @sessionValue</span>
    <div class="d-flex">
        <a class="btn btn-outline-dark btn-sm ms-1" href="/Users/Logout" role="button">Logout</a>
    </div>
}
```

- הוסף דף Logout ריק, שכל תפקידו לבצע מחיקה של Session וחזרה לדף הראשי.

```
public IActionResult OnGet()
{
    // Remove a specific session key
    //HttpContext.Session.Remove("Logout");
    // Clear the entire session
    HttpContext.Session.Clear();

    return RedirectToPage("/Index");
}
```


GM
Internet Programing
Courses


קריית החינוך
השש שנתית
פארק המדע
עתיד של חדשנות

סיכום

סיכום

- כתיבת קוד C# בצד השרת:
  - כתיבת קוד בדף View.
  - כתיבת קוד בקובץ cs (Code behind).
- פעולות OnGet, OnPOst
- אובייקטים ViewData, Session
- העברת מידע מהשרת לדף.
- העברת מידע מהדף (הטופס) לשרת.
- דוגמה – אבטחת דפים.