

תכנות באינטרנט

Backend

Razor pages

גלעד מרקמן

קריית החינוך פארק המדע,

נס ציונה



ASP.NET Core



Stateless Programming

Session Object

- 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.

- 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.
- גם האובייקט ViewData נמחק בכל request והמידע בו לא נשמר.
- נדגים זאת באמצעות דף פשוט המנסה לבצע מונה.

- נבנה טופס פשוט הכולל תיבת טקסט אחד וכפתור. המטרה שבכל לחיצה של הכפתור המונה יגדל באחד.
- לא עובד !!!

```
public class Stateless1Model : PageModel
{
    public int counter { get; set; }

    public void OnGet()...
    public void OnPost()
    {
        counter++;
    }
}
```



```
<form method="post">
  <div class="mb-3">
    <label for="counter" class="form-label">counter</label>
    <input type="text" class="form-control" id="counter" name="counter" value=@Model.counter>
  </div>
  <button type="submit" class="btn btn-primary">Add</button>
</form>
```

- שינוי הקוד תוך שימוש ב Binding פותר את הבעיה בדף הנוכחי.
- אבל פתיחת דף חדש זהה מתחילה את המספור מחדש. האתר לא זוכר את המונה.

```
public class stateless2Model : PageModel
{
    [BindProperty]
    public int counter { get; set; }

    public void OnGet()...
    public void OnPost()
    {
        counter++;
    }
}
```

Stateless 2

counter

Stateless3

- שימוש ב ViewData לא פותר את הבעיה.
- מקבלים שגיאה שכן האובייקט מתאפס בכל קריאה.

```
public class Stateless3Model : PageModel
{
    public void OnGet()
    {
        ViewData["counter"] = 0;
    }
    public void OnPost()
    {
        ViewData["counter"] = (int)ViewData["counter"] + 1;
    }
}
```

```
<form method="post">
  <div class="mb-3">
    <label for="counter" class="form-label">counter</label>
    <input type="text" class="form-control" id="counter" name="counter" value=@ViewData["counter"]>
  </div>
  <button type="submit" class="btn btn-primary">Add</button>
</form>
```

- 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.

- 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

• מונה גלובלי למשתמש באמצעות אובייקט Session.

```
@{  
    ViewData["Title"] = "Stateless4";  
    int? counter = HttpContext.Session.GetInt32("counter");  
}
```

```
<div class="mb-3">  
    <label for="counter" class="form-label">counter</label>  
    <input type="text" class="form-control" id="counter" name="counter" value="@counter">  
</div>
```

```
public class Stateless4Model : PageModel  
{  
    public void OnGet()  
    {  
        if (HttpContext.Session.Get("counter") == null)  
        {  
            HttpContext.Session.SetInt32("counter", 0);  
        }  
    }  
    public void OnPost()  
    {  
        int? counter = HttpContext.Session.GetInt32("counter");  
        HttpContext.Session.SetInt32("counter", (int)counter + 1);  
    }  
}
```

- אובייקט Session הוא מילון השומר מחרוזות ושלמים בלבד.
- ניתן כל העת להוסיף ערכים חדשים או לשנות ערכים קיימים.
- בדפי Razor Page רגילים, הן בדף cshtml והן בדף cshtml.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. אם רוצים להפוך בחזרה ל int רגיל נבצע המרה (int)num.

- בדפי Razor View שאין להם code behind כמו דף Layout, אנחנו נשתמש באובייקט Context.
- נשים לב שעלינו להוסיף `using Microsoft.AspNetCore.Http`
- 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>
}
```