

Háttéralkalmazások

Webes felület szerver oldali generálása

Simon.Gabor@vik.bme.hu 2022-től
Szabo.Gabor@vik.bme.hu 2021-ig



Automatizálási és
Alkalmazott
Informatikai Tanszék

Disclosure

**Ez az oktatási segédanyag a Budapesti Műszaki
és Gazdaságtudományi Egyetem oktatója által
kidolgozott szerzői mű.
Kifejezett felhasználási engedély nélküli
felhasználása szerzői jogi jogsértésnek minősül.**

A szerző elérhetősége:
Simon.Gabor@vik.bme.hu

(ASP).NET Core alapszolgáltatások

Konfiguráció mgmt.

- Kulcs-érték párok
- Alkalmazásbeállítás források
 - > appsettings.json
 - > környezeti változók
 - > Azure Key Vault /App Configuration
 - > Parancssori argumentumok
 - > saját beállításforrás
 - > egyéb fájlok (pl. .ini)
- A forrásokat sorrendbe állítja
 - > Sorrend szerint olvassa a beállításokat
 - > Ha találkozik egy olyan kulccsal, ami korábban már volt, akkor felülírja az értékét (sorrend szerinti összefésülés)
 - > Ha végigért, előáll az aktuális teljes konfiguráció

Környezet mgmt.

- Egy konkrét speciális beállítás, amit külön figyel a keretrendszer: Environment
- Tipikusan környezeti változóként állítjuk
 - > Kulcs: DOTNET_ENVIRONMENT vagy ASPNETCORE_ENVIRONMENT (ha mindkettőt állítjuk, ez az erősebb)
 - > Érték: nincs megkötve, de pár értékhez (Development, Staging, Production) az ASP.NET Core ad támogatást
 - A Developer Exception Page MW **csak Development módban** aktív
 - > appsettings.json **után** az appsettings.[környezetnév].json-t is felolvasa a konfiguráció részeként

appsettings.json, appsettings.Development.json

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  },
  "AllowedHosts": "*"
}

{
  "Logging": {
    "LogLevel": {
      "Default": "Debug",
      "Microsoft.AspNetCore": "Debug"
    }
  }
}
```

launchSettings.json

- Properties mappában található
- Fejlesztői gépen alkalmazandó indítási beállítások
 - > Szerver telepítéskor nem visszük át a szerverre
- IIS beállítások
- Indítási profilok
 - > Pl. IIS Express indítás + böngésző indítás
 - IIS webszerver konfiguráció
 - > Pl. parancssori indítás + böngésző indítás
 - Kestrel webszerver konfiguráció
- Az indítási profilban környezeti változókat is beállíthatunk
 - > A környezetet (ASPNETCORE_ENVIRONMENT) itt érdemes beállítani
 - > A fejlesztői gépen a nem fejlesztői (pl. éles) környezet beállításait is ki tudjuk próbálni

Beállítások kiolvasása programon belül

- Legfelső szintű kódban
 - > `builder.Configuration.GetSection("Logging")`
- Máshonnan: a DI konténerbe a kiinduló builder regisztrál egy `IConfiguration`-t megvalósító példányt (`conf`), ezt injektáltathatjuk
 - > `conf.GetValue<T>("kulcs")` vagy `conf["kulcs"]`

```
public class TestNumModel : PageModel
{
    private readonly IConfiguration Configuration;

    public TestNumModel(IConfiguration configuration)
    {
        Configuration = configuration;
    }

    public IActionResult OnGet()
    {
        var number = Configuration.GetValue<int>("NumberKey", 99);
        return Content($"{number}");
    }
}
```

Kód forrása: [Configuration in ASP.NET Core | Microsoft Docs](#)

Beállítások kiolvasása programon belül

- Egy szekciót egy C# osztályra is lehet képezni (IOptions<T> és társai)
- Környezet alapján elágazás legfelső szintű kódban
 - > `app.Environment.IsDevelopment/IsProduction`
 - > `builder.Environment.IsDevelopment/IsProduction`

Kód forrása: [Configuration in ASP.NET Core | Microsoft Docs](#)

User Secrets

- Ne tároljunk szenzitív adatot a projekt könyvtárán belül!
 - > Pl. ConnectionString az appsettings.json-ben, ha jelszó is van benne
 - > Git repo-ba bekerülhet
- VS: jobbklikk a projekten -> Manage User Secrets
- Csak fejlesztői gépen és fejlesztői (Development) környezetben
- Létrehoz egy extra appsettings fájlt az OS felhasználó saját könyvtárán belül
 - > GUID a fájlnevében
 - > Ezt a GUID-ot a projekt fájlba írja
 - > Appsettings felolvasási sorrend
 1. appsettings.json
 2. appsettings.[környezetnév].json
 3. user secret beállításfájl (csak Development környezetben)
- Nem fejlesztői gépen adjuk meg indításkor más beállítási forrásból (pl. környezeti változó)

Kód forrása: [Configuration in ASP.NET Core | Microsoft Docs](#)

Szerver oldali renderelés

Szerver oldali renderelés

- A szerver a böngésző számára emészthető HTML-t állít elő *futási időben*
- A HTML-ben elhelyezhetjük az (üzleti) adatok alapján a megfelelő HTML elemeket
- Jellemzően a HTML nyelv és egy (esetleg több) további, szerveren használt programozási nyelv vegyítése történik a HTML sablonban, ami betölti az adatokat a HTML-be
 - > Naiv megoldásként a stringek egyszerű összefűzése HTML-lé is technikailag szerver oldali renderelés
- A HTML elkészülte után opcionálisan pl. egy JavaScript alkalmazás is elindul a böngészőben
 - > Esetleg a szerverrel is végez kommunikációt (pl. AJAX)

JSP

```
<%@ page language="java" contentType="text/html; charset=US-ASCII" pageEncoding="US-ASCII"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"https://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=US-ASCII">
<title>First JSP</title>
</head>
<%@ page import="java.util.Date" %>
<body>
<h3>Hi Pankaj</h3><br>
<strong>Current Time is</strong>: <%=new Date() %>

</body>
</html>
```

<https://www.journaldev.com/2021/jsp-example-tutorial-for-beginners>

PHP

```
<!DOCTYPE html>
<html>
<body>

<h2>PHP is <?php echo "Fun!" ?></h2>
<?php
echo "Hello world!<br>";
echo "I'm about to learn PHP!"; "<br>";
echo "Today is " . date("Y/m/d") . "<br>";
?>

</body>
</html>
```

https://www.w3schools.com/php/php_examples.asp

Ruby on Rails

```
<% if @books.blank? %>
<p>There are not any books currently in the system.</p>
<% else %>
<p>These are the current books in our system</p>

<ul id = "books">
  <% @books.each do |c| %>
    <li><%= link_to c.title, {:action => 'show', :id => c.id} -%></li>
  <% end %>
</ul>

<% end %>
<p><%= link_to "Add new Book", {:action => 'new' }%></p>
```

<https://www.tutorialspoint.com/ruby-on-rails/rails-views.htm>

ASP

```
<html>
<head>
  <title>My First ASP Page</title>
</head>
<body bgcolor="white" text="black">
  <%
    Dim strMessage
    strMessage = "Hello World"
    Response.Write (strMessage)
    Response.Write ("<br>")
    Response.Write ("The time on the server is: " & Time())
  %>
</body>
</html>
```

<https://www.webwiz.net/kb/asp-tutorials/first-asp-page.htm>

ASPX

```
<!-- directives -->
<% @Page Language="C#" %>
<!-- code section -->
<script runat="server">
    private void convertupper(object sender, EventArgs e)
    {
        string str = mytext.Value;
        changed_text.InnerHtml = str.ToUpper();
    }
</script>
<!-- Layout -->
<html>
<head>
<title> Change to Upper Case </title>
</head>
<body>
<h3> Conversion to Upper Case </h3>
<form runat="server">
<input runat="server" id="mytext" type="text" />
<input runat="server" id="button1" type="submit" value="Enter..." OnServerClick="convertupper"/>
<br />
<h3> Results: </h3>
<span runat="server" id="changed_text" />
</form>
</body>
</html>
```

https://www.tutorialspoint.com/asp.net/asp.net_first_example.htm

Server-side scripting megoldások

ASP (*.asp)

ActiveVFP (*.avfp)

ASP.NET (*.aspx)

ASP.NET MVC (*.cshtml)

ColdFusion Markup Language (*.cfm)

Go (*.go)

Google Apps Script (*.gs)

Hack (*.php)

Haskell (*.hs)

Yesod

Java (*.jsp)

JavaServer Pages

JavaScript using Server-side JavaScript (*.ssjs, *.js)

Node.js

Lasso (*.lasso)

Lua (*.lp *.op *.lua)

Parser (*.p)

Perl (*.cgi, *.ipl, *.pl)

CGI.pm

PHP (*.php, *.php3, *.php4, *.phtml)

Python (*.py)

Pyramid, Flask, Django

R (*.rhtml)

rApache

Ruby (*.rb, *.rbw)

Ruby on Rails

SMX (*.smx)

Tcl (*.tcl)

WebDNA (*.dna, *.tpl)

Progress WebSpeed (*.r, *.w)

Bigwig (*.wig)

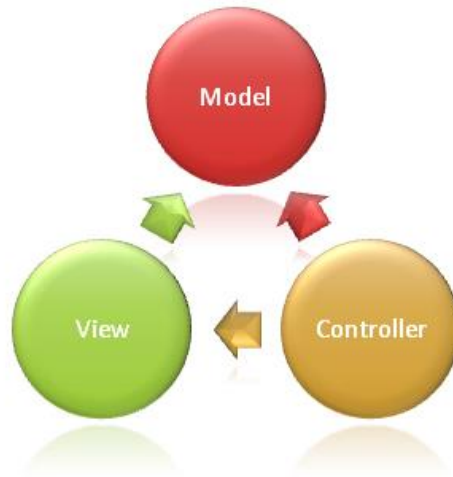
https://en.wikipedia.org/wiki/Server-side_scripting

ASP.NET Core szerver oldali renderelés

- Az alap alkalmazásfelépítés gyakorlatilag megegyezik a szerver- és kliensrenderelt eset között (builder, app, DI, MW, csővezeték)
- Erre az alapra egy UI keretrendszert épít rá
 - > Ennek kimenete lesz a HTML+CSS+JS felület(leírás)
- Két UI keretrendszer
 - > ASP.NET Core Razor Pages
 - Oldal alapú, egy oldalhoz tartozó dolgok egy kupacban, egymás közelében legyenek
 - logika és megjelenítés szétválasztása
 - Projektsablon neve: ASP.NET Core Web App
 - > ASP.NET Core MVC
 - MVC minta szerint szeparál
 - Projektsablon neve: ASP.NET Core Web App (Model-View-Controller)
 - > A HTML sablonokhoz mindkettő ugyanazt a motort használja: **Razor View Engine (RVE)**
 - Az erre vonatkozó diák címe (RVE) végű

MVC

MVC



MVC

- Model
 - > Megjelenítendő adat
 - > ASP.NET Core MVC-nél megegyezhet pl. az entitásmodell egy részével, de lehet külön model réteg is
 - > Nem utazik a hálózaton (közvetlenül)!
- Controller
 - > A kliens által igénybe vehető műveleteket (action-ök) publikál
 - > A modell kezeléséért felelős
 - > Lekérdezés: megszerzi, feltölti az alsóbb réteg (pl. DAL, EF) segítségével
 - > Módosítás: elvégzi a kliens által kezdeményezett módosításokat, frissíti a modellt
 - > Végül átadja valamelyik nézetnek a modellt
 - Műveletenként más-más nézet is lehet
 - Csak ez a pont a lényegi különbség a Web API-s controllerhez képest
 - > A nézet által előállított válasszal tér vissza
- View (nézet) .cshtml
 - > Razor View Engine (ne keverjük össze a Razor Pages-szel!)
 - > Sablon + kódszitek
 - > A model alapján feltölti a sablont, előállítja a kimenetet (HTML+CSS+JS)
 - > A kimenet utazik válaszként a hálózaton

MVC projektsablon - Program.cs

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

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

var app = builder.Build();

// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Home/Error");
}
app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

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

app.Run();
```

MVC kontrolller

- Majdnem ugyanabból az osztályból származik, mint a Web API-s
 - > Controller : ControllerBase
 - > Nézet kezeléssel, meghívással kapcsolatos műveleteket ad a ControllerBase-hez

```
public class HomeController : Controller
{
    private readonly ILogger<HomeController> _logger;
    public HomeController(ILogger<HomeController> logger)
    {
        _logger = logger;
    }
    public IActionResult Index()
    {
        return View();
    }
}
```


Action-ök

- Az Action-ök (akciók) HTTP kérések által végrehajtható, publikus metódusai a kontrollereknek
- Definiálhatják a HTTP igét, az útvonalat, a várt paramétereket
- Visszatérésük lehet:
 - > `IActionResult` vagy ezt megvalósító típus: pl. `StatusCodeResult`, `JsonResult`, amik saját magukat képesek végrehajtani, azaz a válaszbba beleírni
 - > Tetszőleges T objektum: ebben az esetben az objektum sorosításra kerül az alapértelmezett (JSON) vagy megadott sorosítási módszerrel
 - > `ActionResult<T>`: ezzel a visszatérési értékkel a függvényünk T típusú vagy bármilyen `ActionResult` típusú is visszatérhet
 - > A fentiek aszinkron párjai (`Task<IActionResult>`, `Task<T>`, `Task<ActionResult<T>>`, `Task`)
- A visszatérési értéket tipikusan valamilyen Controller őssosztályból örökölt segédfüggvénnyel állítjuk elő

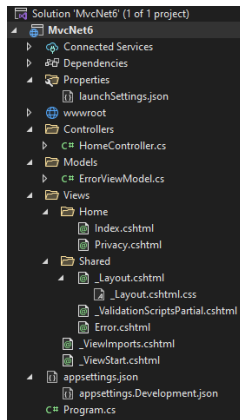
Action segédfüggvények

- A Controller-en, így Action-ból elérhető hasznos függvények.
- Különbőféle ActionResult leszármazottakat állítanak elő
- Controller őszosztály adja
 - > **View**
 - Megkeresi az adott nevű (vagy az Action nevének megfelelő) nézetet (.cshtml) és átadja neki a modellt (opcionálisan)
 - > **PartialView(string viewName, object model)**
 - Egy részleges nézetet ad vissza (layout nélkül)
 - > **Json(object data)**
 - Érdemes a típusos visszatérést preferálni a kézi Json() válaszok helyett
 - > **ViewComponent(Type componentType, object arguments)**
 - Egy ViewComponent hívásának eredményét adja vissza
- ControllerBase őszosztály adja, Web API-ban is használhatjuk
 - > **HTTP státusz kód válaszok**
 - BadRequest(ModelStateDictionary modelState)
 - Created(string uri, object value)
 - NotFound()
 - StatusCode(int statusCode)
 - stb.
 - > **Problem(...)**
 - ProblemDetails szabványnak megfelelő választ állít elő, amit a kliensalkalmazásunk könnyen feldolgozhat
 - > **File(Stream fileStream, string contentType)**

Model binding

- A controllerbe érkező kérések várhatnak paramétert a HTTP kérés alapján
 - > útvonalból (route parameter)
 - > query stringből
 - > fejlécből
 - > törzsből
 - törzsbeli űrlapból
- Web API és MVC esetén is
 - > Attribútumokat teszünk a függvényre és/vagy magára a paraméterre
- Ezek feldolgozását a keretrendszer automatikusan végzi, komplex típusokat is készíthet nekünk
 - > Többféle sorosítást használ, pl. query string, űrlap és törzs más-más módon kerül szabványosan sorosításra

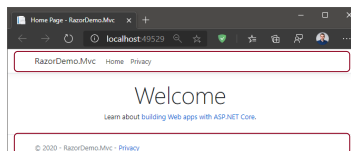
Nézetkiderítés (view discovery)



```
Controllers/HomeController.cs:  
public class HomeController : Controller  
{  
    public IActionResult Index() => View();  
}
```

```
Views/Home/Index.cshtml:  
{  
    ViewData["Title"] = "Home Page";  
}
```

```
<div class="text-center">  
    <h1 class="display-4">Welcome</h1>  
    <p>Learn about <a href="https://docs.microsoft.com/aspnet/core">building  
Web apps with ASP.NET Core</a>.</p>  
</div>
```

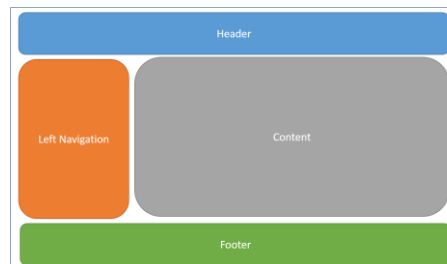


Nézetkiderítés (view discovery)

- A kontrollernek nem kell megmondania közvetlenül, hogy melyik nézet adja a felületet
- Views mappán belül keres konvenció alapján
 - > View("Home ")
- Keresési sorrend
 1. Views/[ControllerName]/[ViewName].cshtml
 2. Views/Shared/[ViewName].cshtml
- Ha semmit nem ad meg (View())
 - > ViewName = Művelet neve

Körítő nézetek (RVE)

- Tipikusan nem csak egy view fájl (.cshtml) vesz részt a válasz előállításában
- A visszaadandó HTML egy jelentős része mindig ugyanaz (fejléc, lábléc, szkripthivatkozások)
 - > Kódduplikálás lenne
- A layout (keret UI) összeállításáért felelnek



_ViewImports.cshtml (RVE)

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

- Hierarchikus hatáskör
 - Minden mellette levő nézet (.cshtml) fájlra érvényes
 - Minden mellette levő mappában levő nézet fájlra érvényes
 - Felüldefiniálás/bővítés a megfelelő hierarchia szinten ugyanilyen nevű fájl létrehozásával
- A hatáskörébe tartozó nézetekbe importál
 - A @using direktívával névtereket importálunk
 - Az @addTagHelper direktívával TagHelper-t tartalmazó assembly-eket töltünk be

_ViewStart.cshtml (RVE)

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

- Hierarchikus hatáskör
- A hatáskörébe tartozó nézetek kódja **előtt** lefut

Layout nézetek (RVE)

- Az `Home/Index.cshtml` layout (elrendezés) oldala a `_Layout.cshtml`
 - Ugyanis hat rá a külső `_ViewStart.cshtml`, ami ezt állítja be
 - A View kódjából is beállíthatnánk/felüldefiniálhatnánk
- A layout oldalak a konkrét oldal renderelése után futnak
- A `RenderBody()` függvényhívás helyére kerül a konkrét nézet kódja
- Egymásba ágyazhatók (layout-nak lehet layout-ja)
- Opcionális vagy kötelező szekciókra hivatkozhat, amiket majd a hozzá tartozó nézetnek kell definiálnia

_Layout.cshtml (RVE)

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>@ViewData["Title"] - RazorDemo.Mvc</title>
  <link rel="stylesheet" href="/lib/bootstrap/dist/css/bootstrap.min.css" />
  <link rel="stylesheet" href="/css/site.css" />
</head>
<body>
  <header>
    <nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-white border-bottom box-shadow mb-3">
      <div class="container">
        <a class="navbar-brand asp-area="" asp-controller="Home" asp-action="Index">RazorDemo.Mvc</a>
        <button class="navbar-toggler" type="button" data-toggle="collapse" data-target=".navbar-collapse" aria-controls="navbarSupportedContent"
          aria-expanded="false" aria-label="Toggle navigation">
          <span class="navbar-toggler-icon"></span>
        </button>
        <div class="navbar-collapse collapse sm-inline-flex flex-sm-row-reverse">
          <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>
          </ul>
        </div>
      </div>
    </nav>
  </header>
  <div class="container">
    <main role="main" class="pb-3">
      <@RenderBody() />
    </main>
  </div>
  <div class="border-top footer text-muted">
    <div class="container">
      <div class="row">
        <div class="col">
          <small>© 2019 - RazorDemo.Mvc - <a asp-area="" asp-controller="Home" asp-action="Privacy">Privacy</a>
        </div>
      </div>
    </div>
  </div>
  <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>
  <@RenderSection("Scripts", required: false) />
</body>
</html>
```

Szerveren futó kódok

Szerveren futó tag helperek

Teljes oldal összeállítása (RVE)

```
Views/ ViewStart.cshtml:
@{
    Layout = "_Layout";
}

Views/Home/Index.cshtml:
@{
    ViewData["Title"] = "Home Page";
}

Views/Shared/_Layout.cshtml:
<DOCTYPE html>
<html lang="en">
<head><title>@ViewData["Title"] - RazorDemo.Mvc</title></head>
<body>
    <header>@* komment *@</header>
    <div class="container">
        <main role="main" class="pb-3">
            @RenderBody()
        </main>
    </div>
    <footer class="border-top footer text-muted">@* komment *@</footer>
    @RenderSection("Scripts", required: false)
</body>
</html>
```

Beállított érték kiolvasása

Behelyettesítés

A nézet nem definiált Scripts nevű, opcionális szekciót, ezért üres lesz

MVC Modell

- A kontroller adhatja át a nézetnek
- Maga a nézet is módosíthatja
- Lehet erősen típusos ...

- Tipikusan az adatmodell típusai jelennek meg
- Mi adjuk meg a típust
- A kontroller View() hívás paramétereként adja át
- Nézet tetején @model direktíva
- A nézet kódjában @Model-ként hivatkozhatunk rá

```
@model ErrorViewModel
{
    if (Model.ShowRequestId) {
        <p>
            <strong>Request ID:</strong> <code>@Model.RequestId</code>
        </p>
    }
}
```

- ... vagy gyengé(bbe)n típusos

- Általánosabb típusú kollektívák, amikhez a kontroller és a nézet is hozzáfér
- Eleve rendelkezésre állnak a nézetben és a kontrollerben, nem kell őket létrehozni, deklarálni, stb.
- ViewData ~ Dictionary<string, object>
- ViewBag ~ dynamic
 - bármilyen property-t beállíthatunk, kiolvashatunk
 - Nekünk kell rá figyelni, hogy olyan property-t olvassunk ki a nézetben, amit beállítottunk a kontrollerben
- ViewData attribútum: kontroller property-kre rakhatjuk.
 - Nézetben kiolvásás: @ViewData["property név"]

Tag Helpers

- Csak szerver oldalon értelmezhető HTML-beli elemek (attribútumok)
- Segítségükkel az aktuálisan rajzolt HTML elemeket bővíthetjük ki/módosíthatjuk
- Számos [beépített](#), de sajátot is írhatunk
- A HTML válasz előállításakor lecserélődnek böngésző által értelmezhető HTML elemekké

```
<a asp-controller="Speaker"
    asp-action="Detail"
    asp-route-id="@Model.SpeakerId">SpeakerId: @Model.SpeakerId</a>
```

↓

```
<a href="/Speaker/Detail/12">SpeakerId: 12</a>
```

HTML Helpers

- Nézet sablonba kényelmesen beágyazható C# függvényhívás, ami a HTML válasz előállításakor lecserélődik HTML részletre
- Számos beépített, de sajátot is írhatunk

```
@Html.ActionLink("Create New", "Create")
```



```
<a href="/Student/Create">Create New</a>
```

MVC – komplexebb példa

Hallgató létrehozás

1. Hallgató létrehozás űrlap megjelenítése (üresen)
 - HTTP GET <http://host:port/Students/Create> => HTML űrlap
 1. Kontrollér művelet a GET /Students/Create -re
 - Más bemenet nem kell
 - meghívja a lenti nézetet, model nincs
 2. Nézet az űrlaphoz
2. Felhasználó kitölt, mentés gombot nyom az űrlapon. Ekkor hallgató beszúrás adatbázisba, aztán hallgatók listájának megjelenítése
 - HTTP POST <http://host:port/Students/Create> => HTTP 302 átirányítás a listázó oldalra
 1. Kontrollér művelet a POST /Students/Create -re
 - bemenet hallgató adat
 - beszúrja a hallgató adatát
 - ha sikerül, átirányít a hallgatók listájának megjelenítése **műveletre** (GET /Students)
 - HTTP GET <http://host:port/Students> => hallgatók listája HTML-ben
 - ha nem sikerül, a fenti űrlapos nézetet hívja kitöltött adatokkal
 2. Kontrollér művelet a GET /Students -re
 3. Nézet hallgatók listájának megjelenítésére

Forrás:

<https://github.com/dotnet/AspNetCore.Docs/tree/main/aspnetcore/data/ef-mvc/intro/samples/cu-final> (nem .NET 6-os!)

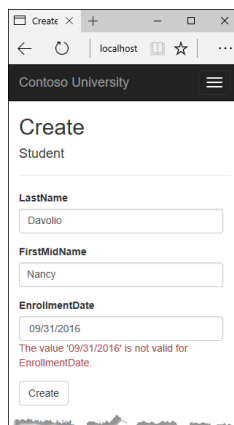
Beszúrás űrlap kérése művelet

```
public class StudentsController : Controller
{
    //....

    // GET: Students/Create
    public IActionResult Create()
    {
        return View();
    }
}
```

Beszúrás űrlap nézet (Students/Create.cshtml)

<https://github.com/dotnet/AspNetCore.Docs/blob/main/aspnetcore/data/ef-mvc/intro/samples/cu-final/Views/Students/Create.cshtml>



The screenshot shows a web browser window with the following content:

- Browser title: Create
- Address bar: localhost
- Page header: Contoso University
- Section title: Create
- Form label: Student
- Form fields:
 - LastName**: Input field containing "Davolio"
 - FirstName**: Input field containing "Nancy"
 - EnrollmentDate**: Date input field containing "09/31/2016"
- Validation error: "The value '09/31/2016' is not valid for EnrollmentDate."
- Submit button: "Create"

Beszűrás művelet

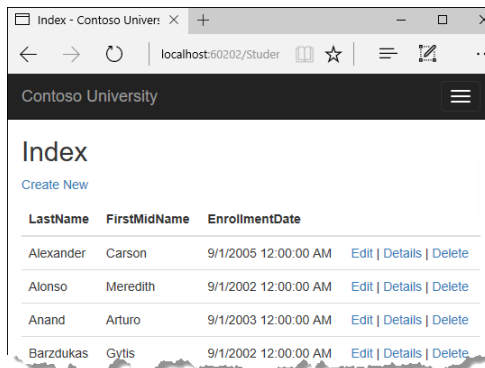
<https://github.com/dotnet/AspNetCore.Docs/blob/096bbbde59f158029fa5e793509f0bf39da2e256/aspnetcore/data/ef-mvc/intro/samples/cu-final/Controllers/StudentsController.cs#L104>

Hallgató listázás művelet

<https://github.com/dotnet/AspNetCore.Docs/blob/096bbbde59f158029fa5e793509f0bf39da2e256/aspnetcore/data/ef-mvc/intro/samples/cu-final/Controllers/StudentsController.cs#L23>

Hallgatólista nézet (Students/Index.cshtml)

<https://github.com/dotnet/AspNetCore.Docs/blob/main/aspnetcore/data/ef-mvc/intro/samples/cu-final/Views/Students/Index.cshtml>



Razor Pages

Razor Pages vs MVC

- A Razor Pages egy modernebb, ugyanakkor régebbi megközelítés
- „Oldalközpontú”
- Egy page , két fájl: a nézet (.cshtml) és a „code behind” (.cs, logika + modell)
 - > Nem kell megtalálniuk egymást, eleve egy egység
 - > Nem kell meghívni a nézetet, a nézet eleve a code-behind adatokból dolgozik
 - > A nézet itt is Razor View Engine alapú, mint MVC-ben
 - Pl. szintaxis, layout logika ugyanaz
- A code-behind tölti be a kontroller szerepét
 - > Kezeli a modellt
 - > PageModel osztályból származik
 - ControllerBase-hez hasonló segédfüggvények
 - > A kliens kérések hozzá futnak be
 - > Előkészíti az adatokat a .cshtml sablonnak
- Lehet egy projektben vegyíteni a két megközelítést
 - > Egy kontroller át tud irányítani Razor Page-re és vice versa

Razor projektsablon - Program.cs

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

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

var app = builder.Build();

// Configure the HTTP request pipeline.
if (!app.Environment.IsDevelopment())
{
    app.UseExceptionHandler("/Error");
}
app.UseStaticFiles();

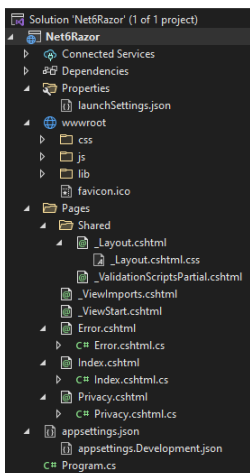
app.UseRouting();

app.UseAuthorization();

app.MapRazorPages();

app.Run();
```

Razor Pages kiinduló projekt felépítés



- Nincs Models/Views/Controller mappa
- A Pages mappa a szokásos nézeteket tartalmazza
 - > A konkrét nézetek mögött van egy-egy `.cshtml.cs` „code behind” fájl

Index oldal

```
Pages\Index.cshtml:
@page
@model IndexModel
@{
    ViewData["Title"] = "Home page";
}

<div class="text-center">
    <h1 class="display-4">Welcome</h1>
    <p>Learn about <a href="https://docs.microsoft.com/aspnet/core">building Web apps with ASP.NET Core</a>.</p>
</div>
```

A `@page` direktíva jelzi, hogy ez egy Razor Page oldal
Opcionálisan megadható, hogy melyik URL útvonalon aktiválódjon

A Razor Page modellje saját maga

```
Pages\Index.cshtml.cs
public class IndexModel : PageModel
{
    private readonly ILogger<IndexModel> _logger;

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

    public void OnGet()
    {
    }
}
```

Ilyen konvenciók segítségével mondjuk meg, hogy az oldal elérhető a GET igével

`void` visszatéréssel a függvény lefutása után kirajzolódik a nézet

Razor Pages routing

- Az URL `http://host:port` utáni részét vetjük össze a projekt `/Pages` mappájában szereplő oldalakkal
 - > `/Students/Create` -> `/Pages/Stundents/Create.cshtml`
- Ha ilyen nincs akkor úgy veszi, mintha az URL végén **/Index** lenne
 - > `/Students` -> `/Pages/Stundents/Index.cshtml`

Fájl helye	Illeszkedő URL
<code>/Pages/Index.cshtml</code>	<code>/</code> or <code>/Index</code>
<code>/Pages/Contact.cshtml</code>	<code>/Contact</code>
<code>/Pages/Store/Contact.cshtml</code>	<code>/Store/Contact</code>
<code>/Pages/Store/Index.cshtml</code>	<code>/Store</code> or <code>/Store/Index</code>

Razor Pages adatkötés

- A `@model` a code-behind osztály, az ő adatai a modell adatai
 - > `[BindProperty]`: a normál modellekötés mechanizmus kitölti a paramétert/tulajdonságot
 - > `[BindProperties]`: az összes tulajdonságot megpróbálja kötni
 - > Két irányú kötés is támogatott
 - A művelet elvégzése előtt: a HTTP kérés alapján kitöltjük a property-t
 - A művelet elvégzése után: a .cshtml innen veszi a válaszhoz az adatot
 - > Válasz HTML előállítás
 - Void-os művelet végén implicit előáll az aktuális állapot alapján a válasz HTML
 - Nem void-os műveleteknél: `Page()` függvény hívásával
- `ViewData` itt is használható, de `ViewBag` nincs

Razor Pages – komplexebb példa

Hallgató létrehozás

1. Hallgató létrehozás űrlap megjelenítése (üresen)
 - HTTP GET <http://host:port/Students/Create> => HTML űrlap
 1. Oldal és bele művelet a GET /Students/Create -re
 - Más bemenet nem kell
2. Felhasználó kitölt, mentés gombot nyom az űrlapon. Ekkor hallgató beszúrás adatbázisba, aztán hallgatók listájának megjelenítése
 - HTTP POST <http://host:port/Students/Create> => HTTP 302 átirányítás a listázó oldalra
 1. Az előző pont meglévő oldalába új művelet a POST /Students/Create -re
 - bemenet hallgató adat
 - beszúrja a hallgató adatát
 - ha sikerül, átirányít a hallgatók listájának megjelenítése **oldalra** (GET /Students)
 - HTTP GET <http://host:port/Students> => hallgatók listája HTML-ben
 - ha nem sikerül, a fenti űrlapos nézetet hívja kitöltött adatokkal (nem valósították meg ☹)
 2. Oldal és bele művelet a GET /Students -re

Forrás:

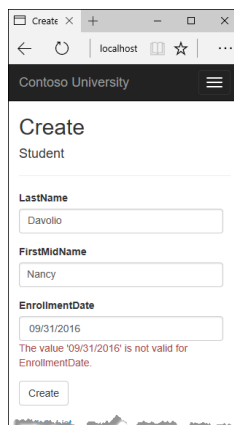
<https://github.com/dotnet/AspNetCore.Docs/blob/main/aspnetcore/data/ef-rp/intro/samples/cu60> (.NET 6-os!)

Beszúrás űrlap kérése művelet

<https://github.com/dotnet/AspNetCore.Docs/blob/096bbbde59f158029fa5e793509f0bf39da2e256/aspnetcore/data/ef-rp/intro/samples/cu60/Pages/Students/Create.cshtml.cs#L22>

Beszúrás űrlap nézet (Students/Create.cshtml)

<https://github.com/dotnet/AspNetCore.Docs/blob/main/aspnetcore/data/ef-rp/intro/samples/cu60/Pages/Students/Create.cshtml>



The screenshot shows a web browser window with the title 'Create' and the URL 'localhost'. The page content is for 'Contoso University' and is titled 'Create Student'. It features three input fields: 'LastName' with the value 'Davolio', 'FirstName' with the value 'Nancy', and 'EnrollmentDate' with the value '09/31/2016'. Below the 'EnrollmentDate' field, there is a red error message: 'The value '09/31/2016' is not valid for EnrollmentDate.' A 'Create' button is located at the bottom of the form.

Beszűrés művelet

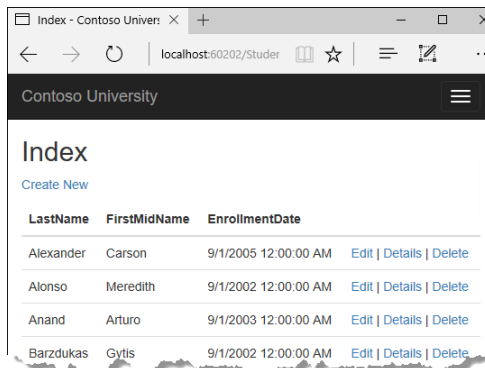
<https://github.com/dotnet/AspNetCore.Docs/blob/096bbbde59f158029fa5e793509f0bf39da2e256/aspnetcore/data/ef-rp/intro/samples/cu60/Pages/Students/Create.cshtml.cs#L32>

Hallgató listázás művelet

<https://github.com/dotnet/AspNetCore.Docs/blob/main/aspnetcore/data/ef-rp/intro/samples/cu60/Pages/Students/Index.cshtml.cs>

Hallgatólista nézet (Students/Index.cshtml)

<https://github.com/dotnet/AspNetCore.Docs/blob/main/aspnetcore/data/ef-rp/intro/samples/cu60/Pages/Students/Index.cshtml>



Kitekintő:
Blazor



```

<h1>Counter</h1>
<p>Current count: @currentCount</p>
<button class="btn btn-primary" @onclick="IncrementCount">Click me</button>

@code {
    private int currentCount = 0;
    private void IncrementCount()
    {
        currentCount++;
    }
}

```

Interactive web UI with C#

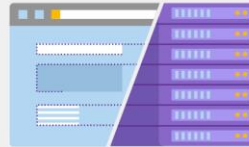
Blazor lets you build interactive web UIs using C# instead of JavaScript. Blazor apps are composed of reusable web UI components implemented using C#, HTML, and CSS. Both client and server code is written in C#, allowing you to share code and libraries.

Blazor is a feature of [ASP.NET](#), the popular web development framework that extends the [.NET developer platform](#) with tools and libraries for building web apps.

Run on WebAssembly or the server

Blazor can run your client-side C# code directly in the browser, using WebAssembly. Because it's real .NET running on WebAssembly, you can re-use code and libraries from server-side parts of your application.

Alternatively, Blazor can run your client logic on the server. Client UI events are sent back to the server using SignalR - a real-time messaging framework. Once execution completes, the required UI changes are sent to the client and merged into the DOM.



<https://dotnet.microsoft.com/apps/aspnet/web-apps/blazor>

Háttéralkalmazások

Webes felület szerver oldali generálása

Szabo.Gabor@vik.bme.hu



Automatizálási és
Alkalmazott
Informatikai Tanszék