УНИВЕРЗИТЕТ У НОВОМ САДУ

ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА У НОВОМ САДУ

CLOUD COMPUTING U ELEKTROENERGETSKIM SISTEMIMA

Osnove *Microsoft WindowsAzure* servisa
Web role

-TEORIJA-

Novi Sad, 2020

Vežba 3 – Windows Azure – Web role

Web role – role namenjenje za prezentacioni sloj u obliku web aplikacija koje se izvšavaju u okviru IIS-a. Osnovna razlika između Worker i Web role je u tome što Web rola pakuje web aplikaciju, postavlja na IIS i nakon čega se automatski servira kroz IIS.

Web rola takođe nasleđuje RoleEntryPoint klasu.

```
0 references
public class WebRole : RoleEntryPoint
{
    0 references
    public override bool OnStart()
    {
        // For information on handling configuration changes
        // see the MSDN topic at https://go.microsoft.com/fwlink/?LinkId=166357.

        return base.OnStart();
    }
}
```

Web role su dodatni omotač na .NET Framework Web aplikacije kako bi mogle da se izvršavaju u okviru Azure Cloud virtualnih mašina.

Jedan od tipova .NET Framework aplikacija je MVC (Model-View-Controller) i biće predstavljen u okviru vežbe. Ovaj tip web aplikacije je MPA (Multiple Pages Application) što znači da kada je potrebno da se promeni stranica na web sajtu mi dobijemo HTML stranice sa Javascript kodom.

HTML (koji nam se prikazuje i koji nam omogućava interakciju kroz forme, linkove i slično) + JS (javascript kod koji se izvršava na našem browser-u, upravlja našim HTML-om, omogućava interakciju sa stranicom u browser-u)

Kontroleri

Korisnici (preko HTML-a i JS koji se izvšava u pretraživaču) interaguju sa kontrolerima. Kada pravimo kontroler u MVC aplikaciji kontroler MORA da se završava sa Controller u svom nazivu. Isto tako nije preporučljivo upotreba donjih crta i posebnih znakova u nazivu. Sve što ide pre sufiksa Controller će se smatrati putanjom do tog kontrolera.
http://localhost:59231/**Student** **– StudentController**

Sve metode koje definišemo u kontroleru su dostupne preko naziva metode u zahtevu.

http://localhost:59231/**Student/GetAllStudents** **– kada stigne ovakav zahtev aplikacija će pokušati da pozove metodu GetAllStudents u StudentController klasi.**

Kontroleri su definisani po REST arhitekturi (RESTful API) gde je svaki zahtev odgovarajućeg tipa i tipovi odgovaraju svim potrebnim za komunikaciju. Tipovi zahteva su **GET, POST, PUT, DELETE, PATCH**.

Sve metode u kontrolerima su podrazumevano REST **GET** metode. **GET** zahtev ne dozvoljava slanje dodatnih resursa na server i očekuje se da su immutable (ne očekuje se promena stanja podataka u bilo kom obliku na serveru).

U našem primeru ovo bi se koristilo za dovlačenje liste studenata ili informacija o jednom studentu i slično.

POST metoda omogućava da se pošalje dodatni resurs kao paket u okviru zahteva. Obrada podataka može da rezultuje promenom stanja na serveru. Npr. dodali smo novog studenta.

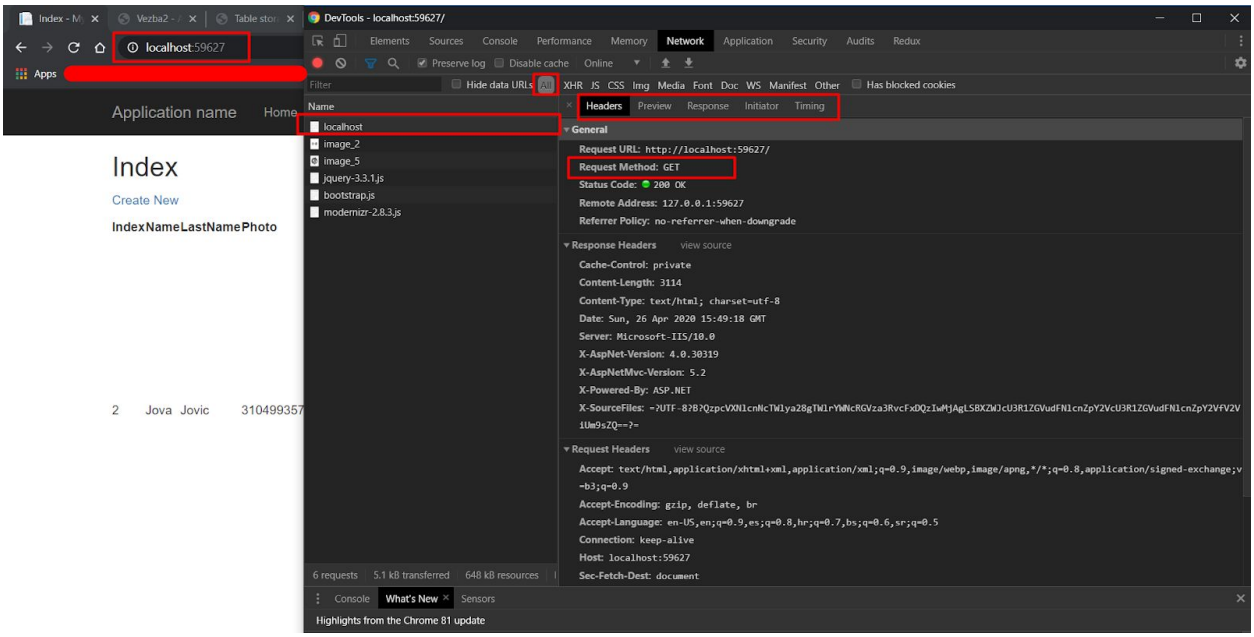
Index metoda u kontroleru je podrazumevana metoda kontrolera tako da će se pozvati kada je zahtev http://localhost:59231/**Student/Index**, ali isto tako I kada je zahtev http://localhost:59231/**Student/**.

Naravno ovo je konfigurabilno, kao što je konfigurabilno koji kontroler preuzima root zahtev odnosno http://localhost:59231/ - ovaj zahtev u sebi nema Student niti metodu. Da bismo definisali kontroler koji podrazumevano preuzima ovaj tip zahteva to možemo da uradimo u RouteConfig klasi.

Komunikacija može da se prati u okviru pretraživača DevTools (F12) – network tab (filter samo XHR zahtevi).

Ovde smo definisali pod defaults stavkom da je podrazumevani kontroler Student kontroler, a da je podrazumevana akcija Index.

Svaki put kada se učita nova putanja, dobija se HTML + JS, tako da kada otvorimo DevTools (F12 na stranici) – network – filter: All videćemo da se prilikom učitavanja stranice šalje GET zahtev ka serveru



Pod Preview/Response možemo da vidimo zaista celu našu HTML stranicu JS kod koji će se u pozadini izvršavati i omogućiti interakciju.

Sve metode MVC-a vraćaju ActionResult što je najčešće View (HTML), PartialView (samo deo HTML-a koji ubacujemo u već postojeću stranicu) ili RedirectToAction redirekcija na drugu akciju kontrolera koji vraća View ili PartialView. View nam vraća odgovarajući HTML iz foldera-a Views. Ukoliko želimo da napravimo studenta idemo na stranicu za pravljenje novog studenta. Create (metoda koja se poziva prilikom klika na Create new student link da bismo došli prvo do stranice sa formom) vraća AddEntity VIEW.

```
public ActionResult Create()
{
    return View("AddEntity");
}
```

Ovaj View mora da se nalazi u okviru Views foldera pod folderom odgovarajućeg kontrolera. U našem slučaju Views – Student – AddEntity. U ovom fajlu nam se nalazi HTML koji ćemo da vratimo.

Ukoliko bismo imali samo `return View();` tada bi aplikacija pokušala da vrati HTML pod nazivom **Create.cshtml** jer nam se tako naziva metoda, što bi rezultovalo greškom.

.cshtml – CSharp HTML

Koristi RAZOR sintaksu kako bi mogao da se korsti C# kod za generisanje HTML-a. Kako to zapravo izgleda na primeru dovlačenja studenata.

Listing 1 – Student/Index.cshtml

```
@model IEnumerable<StudentService_Data.Student>
@{
    ViewBag.Title = "Index";
}
<h2>
    Index</h2>
<p>
    @Html.ActionLink("Create New", "Create")
</p>
<table>
    <tr>
        <th>
            Index
        </th>
        <th>
            Name
        </th>
        <th>
            LastName
        </th>
        <th>
            Photo
        </th>
        <th>
        </th>
    </tr>
    @foreach (var item in Model)
    {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => item.RowKey)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.Name)
            </td>
            <td>
                @Html.DisplayFor(modelItem => item.LastName)
            </td>
            <td>
                <img src="@item.ThumbnailUrl" />
            </td>
            <td>
                @Html.ActionLink("Edit", "Edit", new { /* id=item.PrimaryKey */ })
                @Html.ActionLink("Details", "Details", new { /* id=item.PrimaryKey */ })
                @Html.ActionLink("Delete", "Delete", new { /* id=item.PrimaryKey */ })
            </td>
        </tr>
    }
</table>
```

Prvo je definisanje modela `@model Ienumerable<StudentService_Data.Student>.`

Ovo je ulazni parametar koji će View() očekivati.

```
return View(StudentDataRepository.RetrieveAllStudents());
```

Tako da kada krene da se gradi HTML podatke koje će dobiti jesu studenti iz metode RetrieveAllStudents. C# kod počine sa @.

```
@{
    ViewBag.Title = "Index";
}
```

ViewBag je poput globalno dostupnog objekta na koje možemo da kačimo dodatne informacije. Ima dinamičke atribute tako da možemo da u metodi kontrolera nakačimo npr. `ViewBag.Error = "Some error";`

I da ovako nešto evaluiramo u .cshtml-u.

```
@{
    if(ViewBag.Error == "SomeError")
    {
        <p>Some error occured</p>
    }
}
```

Izraz se evaluira i ukoliko je zadovoljen, generisaće se i PARAGRAF tag u HTML-u sa tekstom SOME ERROR OCCURED i biće deo HTML-a koji se vraća korisniku.

HTML helpers

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

HTML helpers su ništa više nego statičke C# metode koje vraćaju neki HTML izmenjen u skladu sa prosleđenim parametrima. Npr. helper koji je prikazan na početku zapravo vraća HTML u ovom obliku `<a href="/Student/Create">Create New</a>`.

Ukoliko koristite više puta jedan te isti HTML npr. header tabela za prikaz studenata i tabela za prikaz nastavnika. Izgled i HTML je isti, samo se razlikuju nazivi. Možemo da napravimo helper koji nam ovo generiše, a mi samo prosledimo kolone u obliku liste naziva. Za više informacija kako praviti HTML helpere možete da viditi u .NET Framework dokumentaciji pravljenje CUSTOM helper-a.

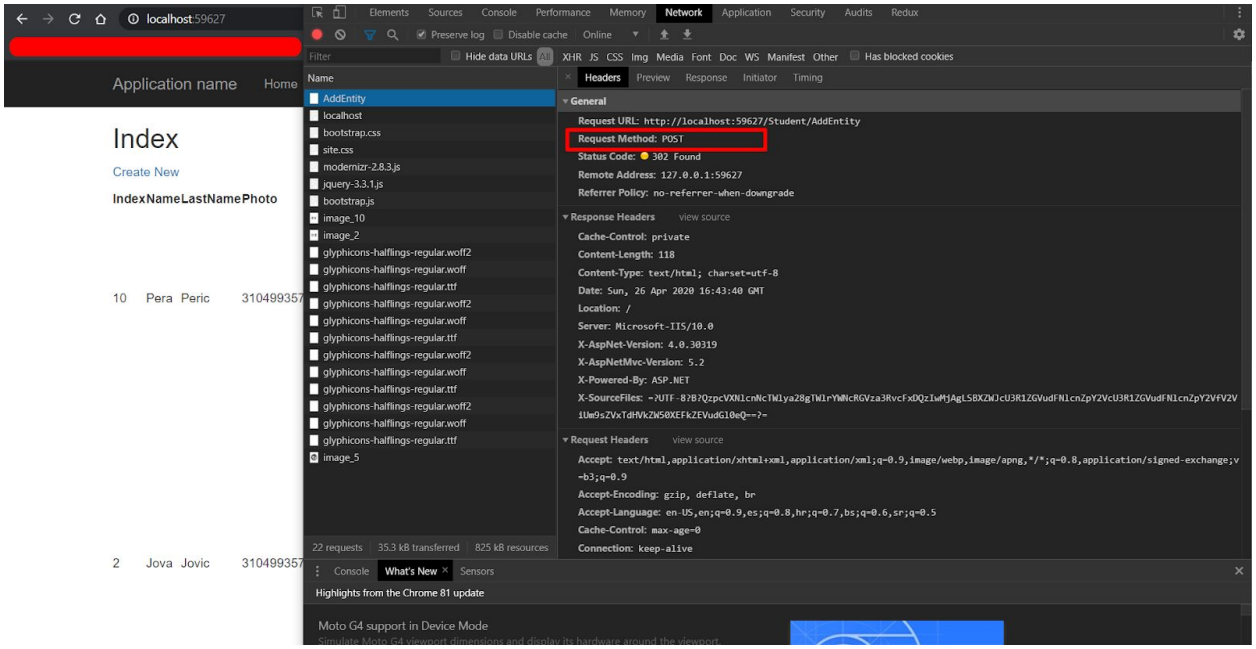
HTML forma

```
@using (Html.BeginForm("AddEntity", "Student", FormMethod.Post, new { enctype =
"multipart/form-data" }))
{
    @Html.ValidationSummary(true)
    <fieldset>
        <legend>Student</legend>
        <div class="editor-label">
            @Html.LabelFor(model => model.RowKey)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.RowKey)
            @Html.ValidationMessageFor(model => model.RowKey)
        </div>
        <div class="editor-label">
            @Html.LabelFor(model => model.Name)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.Name)
            @Html.ValidationMessageFor(model => model.Name)
        </div>
        <div class="editor-label">
            @Html.LabelFor(model => model.LastName)
        </div>
        <div class="editor-field">
            @Html.EditorFor(model => model.LastName)
            @Html.ValidationMessageFor(model => model.LastName)
        </div>
        <input type="file" id="studentPicture" name="file" />
        <p>
            <input type="submit" value="Create" />
        </p>
    </fieldset>
}
<div>
    @Html.ActionLink("Back to List", "Index")
</div>
```

HTML koji se generiše i dostavlja pretraživaču izgleda **slično** ovome:

```
<form action="/Student/AddEntity" method="POST">
    <label for="Name">First name:</label><br>
    <input type="text" name="Name"><br>
    <label for="LastName">Last name:</label><br>
    <input type="text" name="LastName"><br><br>
    <input type="submit" value="Submit">
</form>
```

Klikom na dugme `<input type="submit" value="Submit">` šaljemo **POST** zahtev sa informacijama o studentu.



MODELS

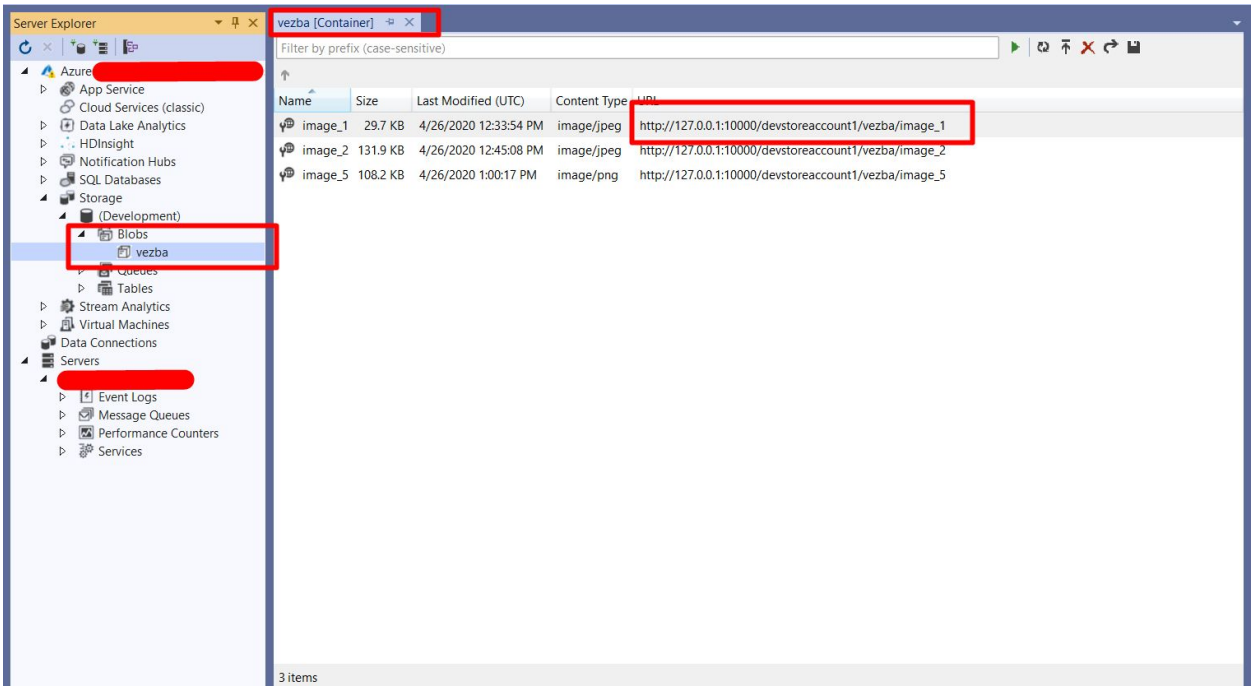
```
[HttpPost]
0 references
public ActionResult AddEntity
```

Models – namenjeno pravljenju modela sa kojima će Views da rade. U primeru zadatka je korišćen Student model pa se ne koristi poseban model u **Models** folder-u.

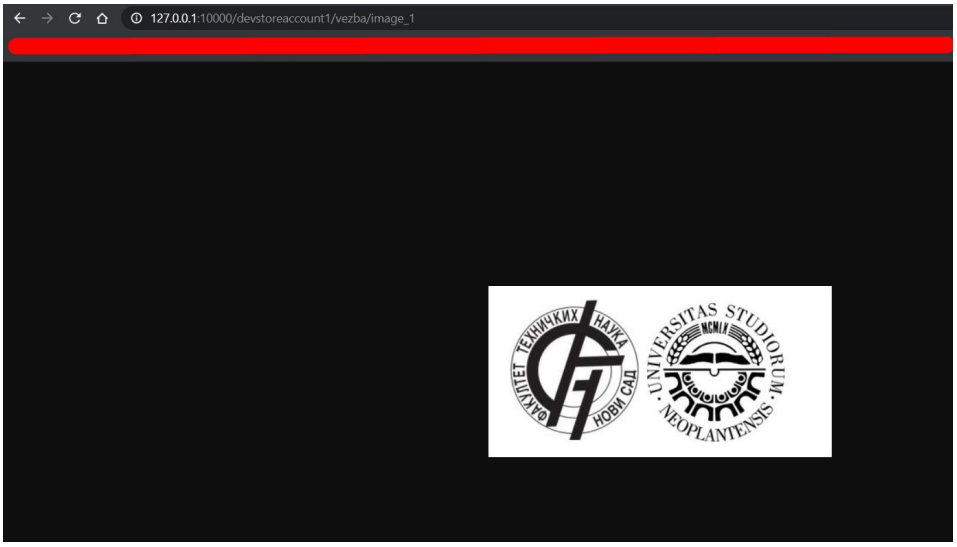
```
@model IEnumerable<StudentsService_Data.Student>
```

Bez obzira na ovo stranica bi po najboljoj praksi trebalo da ima odgovarajući StudentViewModel.cs gde smo mapirali sve ove podatke koji su nam potrebni za našu stranicu. Sve bi to moglo da nasleđuje i neku klasu ViewModel gde bismo kačili properties poput npr. Error kako bismo znali da l da prikažemo neki paragraf sa greškom.

NAPOMENA



U samom zadatku u zabelama čuvamo putanje do resursa koji se nalaze u blob storage-u. Ne čuvamo same slike u tabeli već putanje. Tako da u tabeli čuvamo putanje do slika, a slike su zaista dostupne ako ovaj URL unesemo u pretraživaču.



Razlika koju ćete primetiti prilikom pravljenja isto zahteva za pravljenje **NASTAVNIKA** je u tome što se kod nastavnika vidi Form Data. Iz razloga što kad šaljemo fajl zajedno sa podacima tada se moraju na drugačiji način slati kroz drugi mehanizam i nije vidljivo.