Razvoj Web Aplikacija

Predavanje 12

Dizajn i arhitektura MVC aplikacije

- Jednostavna monolitna arhitektura
- Troslojna arhitektura
- Autentifikacija
- Stranice greške

Troslojna arhitektura

3-slojna arhitektura (2,5-slojna)

- Više projekata
  - **BL** projekt
  - **MVC** projekt – koristi BL projekt
  - **Web API** projekt (treba dodati) – koristi BL projekt
- Radit ćemo s "2,5-slojnom" arhitekturom
  - DAL nema svoj zasebni projekt
  - DAL sloj se realizira kroz DB kontekst
- BL projekt koristi DAL
  - Pristup bazi podataka omogućen je unutar BL projekta
- Postoji više vrsta modela
  - Viewmodel, BL model i DAL model
  - Dobro je imati **Automapper**
  - Automapper se može koristiti za mapiranje u oba smjera, ali se najčešće koristi za mapiranje u smjeru dohvaćanja podataka, a ne u smjeru pohranjivanja podataka
- Koristit ćemo jednostavan **repository**
  - Parametri su primitivnog tipa
  - Vraća BL modele

Troslojna arhitektura - DEMO

- https://www.youtube.com/watch?v=KHy63h7b1As

Troslojna arhitektura - SAŽETAK

- BL projekt – struktura
  - **BLModels** mapa
  - **DALModels** mapa – za "reverse engineered" modele
  - **Mapping** mapa – za Automapper profil (DAL ➔ BL)
  - **Repositories** mapa – za repozitorij (vraća BL model)
- MVC projekt – struktura
  - **Viewmodels** mapa – za viewmodels
  - **Mapping** mapa – za Automapper profil (BL ➔ viewmodel)
  - **Controllers**, **Views**, wwwroot, Properties mape

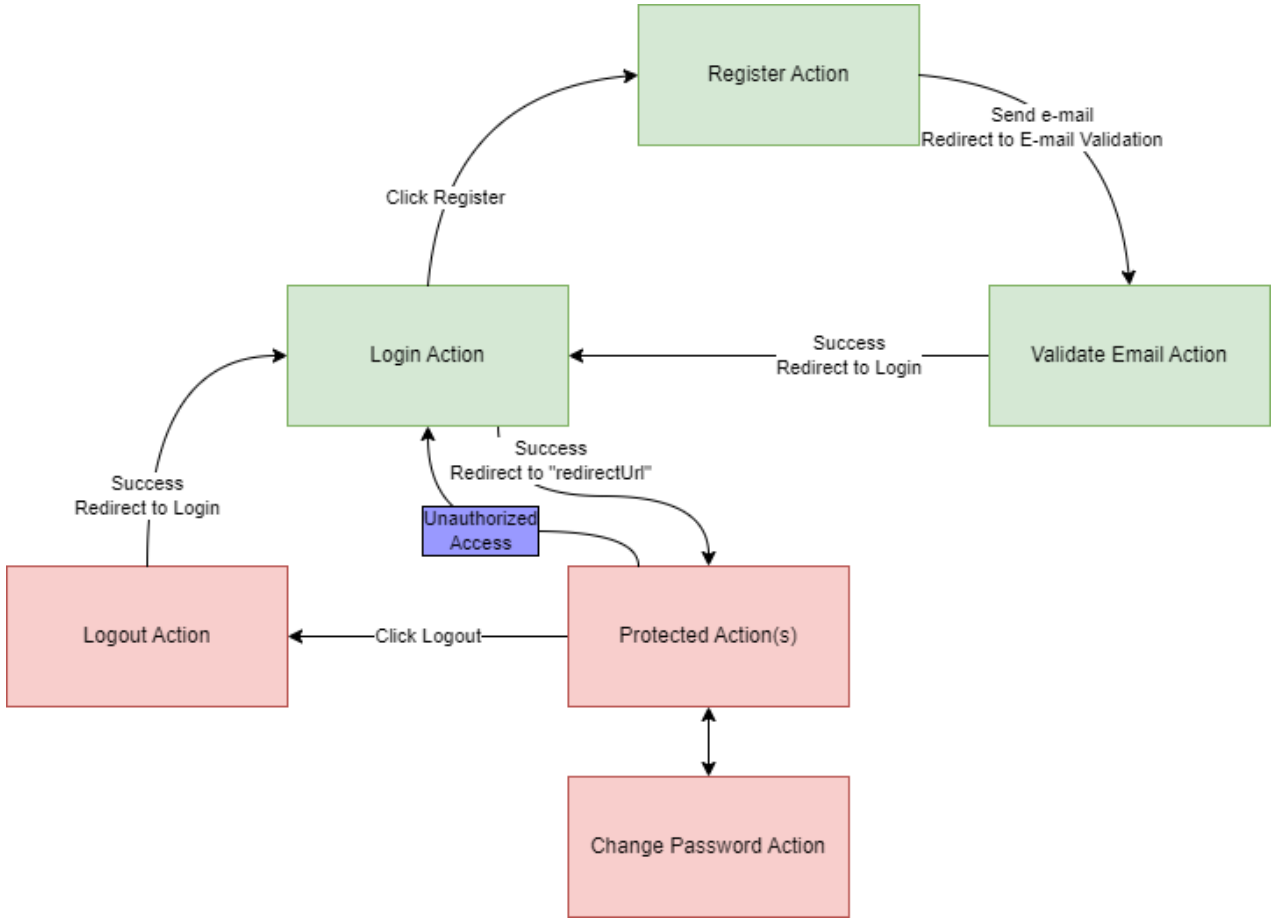
Troslojna arhitektura (nastavak)

- Ova arhitektura omogućuje dodavanje web API aplikacije koja koristi BL projekt na isti način kao što ga koristi MVC projekt
- Neka dodatna poboljšanja, jednostavna za dodavanje:
  - Try/catch blokovi u kacijama kontrolera
  - Stranice grešaka
  - Logiranje
  - …
- Važan koncept koji treba uzeti u obzir: **autentifikacija**

Autentifikacija

Autentifikacija

- Registracija korisnika
  - ➔ `GET+POST User/Register`
- Validacija (provjera valjanosti) adrese e-pošte
  - ➔ `GET User/ValidateEmail`
- Prijava (izdavanje tokena/kolačića)
  - ➔ `GET+POST User/Login`
- Odjava
  - ➔ `POST User/Logout`
- Promjena lozinke
  - ➔ `GET+POST User/ChangePassword`
- Zaštićene akcije



Autentifikacija – koncepti

Akcija	Model	GET	POST	POST preusmjeravanje	View
Register	VMRegister	✓	✓	ValidateEmail	Register.cshtml
ValidateEmail	VMValidateEmail	✓	✗	Login	ValidateEmail.cshtml
Login	VMLogin	✓	✓	**{redirectUrl}** or Index	Login.cshtml
Logout	VMLogout	✗	✓	Login	✗
ChangePassword	VMChangePassword	✓	✓	Index	ChangePassword.cshtml

Autentifikacijski middleware

- Koristit ćemo cookie autentifikaciju (ne JWT)

- Koristit ćemo ugrađeni middleware za autentifikaciju
  - Postaviti u Program.cs
  - `app.UseAuthetication()`
  - `app.UseAuthorization()`

- Middleware zahtjeva auth servise

```
builder.Services
    .AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme)
    .AddCookie()
```

- Upravljanje s auth cookie
  - `HttpContext.SignInAsync()` ➔ stvara a cookie
  - `HttpContext.SignOutAsync()` ➔ uklanja cookie

Autentifikacijski middleware (nastavak)

Što radi auth middleware?

- Ako je ispravno postavljen, za svaku zaštićenu stranicu middleware traži ispravan auth cookie
- Za takvu stranicu, ako middleware ne primi kolačić, vraća status 404 klijentu

Što **NE** radi auth middleware?

- Validacija korisnika – provjeru postoji li korisnik u bazi podataka (s odgovarajućom lozinkom) morate učiniti sami
- Enkripcija/hashing lozinke – da biste zaštitili lozinku od neovlaštenog mijenjanja, morate koristiti kriptografske funkcije
- Rukovanje greškama/iznimkama – morate sami riješiti iznimke (try/catch)

Stranice s prikazom greške

Stranice s prikazom greške

- Odnosi se na slučaj kada se pojavi iznimka te ista nije uhvaćena
- Primjer: ne postoji try/catch unutar akcije
  - Slučaj A: dogoditi će se "default" prikaz greške, prazna stranica sa HTTP statusom - nije "user-friendly"
  - Slučaj B: dogoditi će se prikaz svih podataka greške na temelju kojih sposobni hacker može preciznije odlučivati o sljedećoj malicioznoj akciji
  - Želimo izbjeći oba slučaja
- Prikaz stranica greške svodi se na korištenje ispravnog middleware-a

Stranice s prikazom greške

Referenca

- **UseDeveloperExceptionPage()**
  - ➔ U slučaju neulovljene iznimke prikazuje kompletni kontekst greške
  - ➔ Koristiti **isključivo** za vrijeme razvoja

- **UseExceptionHandler("/Error")**
  - ➔U slučaju neulovljene iznimke preusmjerava zahtjev na URL koji se prosljeđuje kao parametar (ovdje: "/Error")

- **UseStatusCodePages(Text.Html, "Status Code Page: {0}")**
  - ➔ U slučaju statusnog koda greške prikazuje tekst kao što je npr. "Status Code: 404; Not Found"
  - ➔ Moguće je konfigurirati tako da prikazuje HTML, te također prilagoditi tekst

Stranice s prikazom greške

Referenca

- **UseStatusCodePagesWithRedirects("/StatusCode/Error/{0}")**
  - ➔U slučaju statusnog koda greške preusmjerava zahtjev na URL koji se prosljeđuje kao parametar (ovdje: "/StatusCode/Error")
  - ➔Također se kao zadnji dio URL-a dodaje broj statusnog koda (npr. 404), što treba onda konfigurirati kod poziva metode akcije, kao npr. `[RouteAttribute("/[controller]/[action]/{code}")]`

- **UseStatusCodePagesWithReExecute("/StatusCode/Error/{0}")**
  - ➔Kao UseStatusCodePagesWithRedirects(), ali se preusmjeravanje ne radi na klijentu već na serveru
  - ➔U stvari se ponovno izvrši cijeli zahtjev pa se nakon toga pozove akcija na koju je podešeno preusmjeravanje

Hvala na pažnji!