**Elektrotehnička škola “Mihajlo Pupin”**

Novi Sad, Futoška 17



**Seminarski rad iz Veb programiranja**

**ASP.NET Core MVC**

Mentor: Učenik:

Danijela Radmilović Diana i Šantavec, IT42

SADRŽAJ Strana

[3](#__RefHeading___Toc1010_2618295053)

[1. Uvod 4](#__RefHeading___Toc1012_2618295053)

[2. KARAKTERISTIKE 5](#__RefHeading___Toc1014_2618295053)

[2.1. Rutiranje 5](#__RefHeading___Toc1016_2618295053)

[2.2. Povezivanje modela 5](#__RefHeading___Toc1018_2618295053)

[2.3. Validacija modela 5](#__RefHeading___Toc1020_2618295053)

[2.4. Cross platform 5](#__RefHeading___Toc1022_2618295053)

[2.5. Open Source 6](#__RefHeading___Toc1024_2618295053)

[2.6. Testability 6](#__RefHeading___Toc1026_2618295053)

[2.6. Learning Curve 6](#__RefHeading___Toc1028_2618295053)

[3. DRUGOG POGLAVLJA 7](#__RefHeading___Toc1030_2618295053)

[3.1. *MVC pattern* 7](#__RefHeading___Toc1032_2618295053)

[3.2. *Rutiranje* 8](#__RefHeading___Toc1034_2618295053)

[3.2.1. *URL i segmenti URL-a* 8](#__RefHeading___Toc1036_2618295053)

[3.2.2. *Middleware* 8](#__RefHeading___Toc1038_2618295053)

[3.2.3. *Rutiranje* 9](#__RefHeading___Toc1040_2618295053)

[3.3. *Testability* 9](#__RefHeading___Toc1042_2618295053)

[3.3.1 Uint testing 9](#__RefHeading___Toc1044_2618295053)

[*3.3.2. Unit testing u ASP.NET Core MVC-u* 10](#__RefHeading___Toc1046_2618295053)

[*3.4. Pokretanje na Linux-u* 10](#__RefHeading___Toc1048_2618295053)

[4. zaključak 12](#__RefHeading___Toc1050_2618295053)

[Prilog 13](#__RefHeading___Toc1052_2618295053)

[Prilog 1 13](#__RefHeading___Toc1054_2618295053)

[Literatura 14](#__RefHeading___Toc1056_2618295053)

# 1. seminarskiUvod

Originalno MVC okruženje je Microsoft objavio u oktobru 2007. godine i odrazilo se na tadašnje trendove web aplikacija koji su i sami bili u razvoju. MVC okruženje je izgrađeno na postojećoj ASP.NET platformi (objavljena 2002) iz razloga što je ista imala dosta dobrih funkcionalnosti niskog nivoa (odnosno dobru apstrakciju web-a) i koja je bila dobro poznata ASP.NET programerima, te je podržavao napredak u razvoju. Međutim, MVC okruženje je zahtevalo od korisnika neka podešavanja koja uopšte nisu imala efekta na njihove web aplikacije. Još jedan problem je nastao kada je u toku razvoja *Micorsoft* počeo da dodaje poboljšanja i u MVC okruženju, a i da proširuje ASP.NET, pa je došlo do defragmentacije koda. [2]

2015. godine, *Microsoft* je objavio novi pravac razovja ASP.NET i MVC okruženja što je dovelo do nastanka ASP.NET Core MVC. ASP.NET Core MVC je razvojno okruženje za serverski deo aplikacija koje ne iziskuje puno računarskih resursa [1,3]. Što se tiče samog programiranja, naglašava čistu arhitekturu, upotrebu *design patterns*[[1]](#footnote-2)-a i ne prikriva način rada samog Web-a [2]. Pokreće se na ASP.NET [[2]](#footnote-3)platformi što mu omogučava da se aplikacije mogu pokrenuti na različitim platformama (Linux, Mac OS X).

Dalji rad prikazuje neke od karakteristike ovog okruženja, u kratkim crtama objašnjava kako se radi sa *controllers*, *views* i *models*, kako se radi validacija i kako se pravi aplikacija na Linux-u i u poslednjem poglavlju postupak pravljenja jednog jednostavnog primera.

# 

# 2. seminarskiKARAKTERISTIKE

Pre nego što se pređe na sam rad i primer pravljenja aplikacije biće navedene neke važnije karakteristike *ASP.NET MVC Core* radnog okruženja i objašnjenja za svaku.

## 2.1. Rutiranje

Sadrži komponentu za mapiranje URL-ova, što omogućava da bez obzira na organizaciju fajlova na samom serveru, pretraga i linkovi budu ispravo generisani. Omogućava dva tipa rutiranja: *Convention-based routing* (definisanje URL formata koja aplikacija prihvata i dalje sprovođenje zahteva) i *Attribute routing* (definisanje informacija o rutiranju, tako sto se kotroleru i akcijama doda adređen atribut).

## 2.2. Povezivanje modela

Konvertuje podatke, koji čine zahtev klijenta, u objekte koje kontroler može da obradi i na taj način logika kontrolera ne mora da dodatno obrađuje podatke koje prima i kod logike kontrolera je jednostavniji i pregledniji.

## 2.3. Validacija modela

Pre nego što kontroler uzme da obrađuje podatke, isti bivaju provereni što obezbeđuje bezbednost i servera i aplikacije. Ukoliko se otkrije neka nepravilnost, podaci neće biti prosleđeni serveru i kontroleru.

## 2.4. Cross platform

Omogućava da se aplikacija ne mora pokretati na Windows serverima, već može i na serverima sa drugim operativnim sistemima.

## 2.5. Open Source

Ova platforma je otvorenog koda, što omogućava brži razvoj okruženja, sam kvalitet, otkrivanje i rešavanje postojećih grešaka, i slično.

## 2.6. Testability

Okruženje omogućva rad Unit testing-a, što dovodi do toga da se greške uoče na vreme i da se mogu ispraviti. Pored navedene prednosti, u samom nazivu okruženja se nalazi MVC što nam govori da koristi *Model-View-Controller design pattern* (jedna od implementacija separation of concerns principa). Pored preglednosti koda, ovo omogućava i bolju raspodelu rada na projektu.

## 2.6. Learning Curve

Za razliku od asp.NET okruženja, ovo okruženje zahteva više vremena i napora da se nauči.

# 3. DRUGOG POGLAVLJA

## 3.1. ***MVC pattern***

Pre nego što se pređe dalje na sam asp.net core MVC, bitno je razumeti šta zapravo predstavlja *MVC pattern*. Svaki deo ovog pattern-a je dobro definisam i samostalan, što se odnosi na *separation of concerns* *design pattern* (SoC). Sam termin se koristi još od 1970-tih i bio je način da se organizuju neke rane GUI aplikacije. Podrazumeva da se aplikacija deli na tri dela:

* Model – sadrži ili predstavlja podatke sa kojima korisnik radi
* View – obrađuje određen deo modela i predstavlja ga kao korisniči interfejs
* Control – obrađuje zahteve izvršava potrebne operacije nad Models i određuje views koji će prikazati podatke korisniku.

Ciljevi ovog modela su da se razvoj jednostavno podeli i paralelno izvršava i da se određen deo koda može koristiti u više aplikacija. Pored navedenog, još neke prednotsi su: mala povezanost između navedenih delova, jednostavne izmene (zbog SoC-a), jedan model može imati više prikaza, lakše testiranje koda. Neka od mana ovog modela su: programeri moraju da poznaju više tehnologija, UI aplikacije su već izdeljene na delove i onda MVC gubi svoj značaj, novi slojevi otežavaju navigacju kodom.

Samo kreiranje svakog ovog dela je prilično jednostavno.

* Model : Desni klik na Model folder → Add → Class
* View: desni klik na Views folder → Add new folder, onda na desni klik na kreiran folder → Add → New item → Razor view (MVCMovie Dialog)
* Controll: desni klik na Controllers folder → Add → Controller → MVC Controller Empty (Add Scaffold dialog box).

U sledećem poglavlju će biti primer upotrebe i izgleda ovih delova

## 3.2. ***seminarskiRutiranje***

Kao što je već navedeno u prethodnom poglavlju, rutiranje je jedna od prednosti ASP.NET Core MVC-a. Pre nego šte se pređe na samu priču o rutitanju bitno je napomeuti i postojanje *Middleware*-a i šta je sam URL i koji su njegovi segmenti, iz razloga što se napomenuti termini pojavljuju u procesu rutiranja.

### **3.2.1. *URL i segmenti URL-a***

Uniform Resource Locator (URL), poznatije kao web adresa, koristi se da definiše adresu na WWW-u. Protokol je zadužen da odredi na koji način će se podaci sa linka obrađivati. Protokol bitan za dalji tekst je HTTP protokol koji se koristi za web resurse. Pored *hostname*-a, sadrži i segmente koji su razdojeni sa ‚/‘ karakterom. U primeru: <http://www.primer.com/Home/Index>, Home je prvi segment, dok je Index drugi. Broj segmenata nije ograničen, ali većini aplikacija su dovoljna samo tri. Ukoliko se unese segment koji aplikacija ne može da pronađe, pojavljuje se greška 404.

### **3.2.2. *Middleware***

Kontroliše kako aplikacija reaguje na HTTP zahtev i ima veliku ulogu kada je potebna autentifikacija i autorizacija korisnika. Ubcuje se u *pipeline*[[3]](#footnote-4) applikacije da bi rukovao zahtevima i odgovorima. Svaka komponenta određuje da li će proslediti dalje zahtev na sledeću komponentu, ali ima mogućnost da izvrši i neke akcije i pre i posle prosleđivanja podataka. Svaki deo je veoma specifičan objekat fokusiran samo na oređenu ulogu. Da bi aplikacija funkcionisala kako je predviđeno, potrebno je više *middleware*-a.

### **3.2.3. *Rutiranje***

Rutiranje je proces u kom aplikacija povezuje dobijenu URL putanju sa izvršnim *endpoint*-ima aplikacije koje izvršavaju određene akcije, preko *middleware*-a. Dve osnovne funkcijeprocesa rutiranja su:

* mapiranje primnjenog URL-a kontrolorima i akcijama za dalji rad
* generisanje izlazog URL-a u vidu linka korisniku, tako da kada korisnik pokuša da pristupi dobijenom URL-u ne dođe do greške

Aplikacije mogu definisati rute upotrebom, kontrolera, Razor Pages-a, SignalR-a, gRPC Servise-a, i slično. Postoje dva tipa rutiranja:

* *Coventional Routing*
* *Attribute Routing* (Uveden sa MVC5 i omogućava veću fleksibilnost)

## ***3.3. Testability***

Pre samog objašnjenea kako da se *unit* testoviodrade u ASP.NET Core MVC-u, prvo će biti par rečenica šta zapravo znači Unit testing i zbog čega ga treba koristiti.

### 3.3.1 *Unit testing*

*Unit* testing šredstavlja testiranje softvera tako što se izdvoje posebne komponente, odnosno *unit*-i. Sam *unit* predstavlja najmanji moguć deo nekog seftvera koji može biti testiran. U objekno orijentisan programiranju, najmanji *unit* je metod, u proceduralnom programiranju to može biti ili poseban program ili funkcija ili procedura.

Testovi se izvršavaju u toku pisanja koda i vrše ga sami developeri. Predstavlja prvi nivo u testiranju softvera u nekoliko modela (SDLC, STLC, V Model). Otkrivanje greške u nekom od kasnijih faza testiranja (Unit testing, Integration Testing, System Testing i Acceptance Testing) bi dovelo do pitanja gde je zapravo greška, pisanje ovih testova štedi vreme. Dobro ga je primenjivati još i zato što se uz dobro napisane testove, prilikom izmene koda lako može otkriti problem, ako postoji. Kodovi pisani tako da se *unit* tesitranje može izvršiti su modularni i slabo zavise jedni delovi koda od drugih, što dovodi do toga da se mogu više puta primeniti. Vreme za pisanje testa biva nadoknađeno vremenom ispravljanja problema, a i ovakvi testovi su pouzdaniji od npr postavljanja brakepoint-a. Sve i ako program ima grešku kada se ubaci u upotrebu, otklanjanje greške je jednostavnije. Pored prednosti u samom kodu i razvoju aplikacije, ovi testovi potpomažu i pisanje dokumentacije i razumevanje samog koda. Naravno, *unit* testovi ne mogu uhvatiti svaku grešku u kodu, niti greške nastale tokom integracije, što iziskuje pisanje i drugih testova. Testiranje se može raditi ručno i automatski.

### ***3.3.2. Unit testing u ASP.NET Core MVC-u***

Dobro napisana Asp.Net Core MVC aplikacija bi trebala da zahteva mali broj testova, zato što sadrži kontrolere, filtere ,*Views* i *ViewModels*. Jedan značajan koncept koji govori šta je potrebnoodraditi je AAA (Arange, Act, Assert): definisati i/ili inicijalizovati varijable i objekte, pozvati metode i proslediti im parametre i onda proveriti validnost izlaza

Najbrži način za kreiranje testa je korišćenje postojećeg šablona koji pravi prazan test. Proces pravljenja šablona za test nije složen. Prvo je potrebno izvršiti desni klik na Solution → Add → New Project → xUint Test Project (ili MSTest). Sledeći korak je dodati reference: desni klik na dependencies → Add refrence. Kada se i to odradi potrebno je ubaciti NuGet pakete (pružaju neke podatke dostupne u toku izvršavanja) za projekat. Posle toga je potrebno dodati kod za testiranje određenog *unit*-a koji zavisi od same aplikacije.Nakon toga preostaje samo još pokrenuti test i proveriti validnost ispisa.

***3.4. Pokretanje na Linux-u***

Kao što je napomenuto u uvodu, ASP.NET Core MVC je cross platform. U ovom delu teksta će biti objašnjeno kako može da se napravi aplikacija na Linux operativnom sistemu.

Još pre ASP.Net Core MVC, postojalo je Mono okruženje (open source i cross-platform) koje je pokretalo .NET aplikacije na Linux-u. Koristi Apache (Apache HTTP Server) za hostovanje ASP.NET aplikacija. Kako nisu sve BLC biblioteke (.NET Framework base clas library) portovane na Linux, potrebno je koristiti Mono okruženje. Kao web server može da se koristi Kestrel koji je takođe cross-platform i može da pokrene ASP.NET Core kod. Ukoliko aplikacija prima podatke sa interneta, Kestrel nije dovoljan, već je potrebno konfigurisati Apache server kao reverse proxy server da bi preusmerio HTTP zahteve ASP.NET Core aplikaciji. Osim Apache servera može se koristiti i IIS ili Ngixn. Bitniji koraci potrebne konfiguracije Apache servera su omogućavanje modproxy modula i kreiranje konfiguracionog fajla. Nakon što se odrade pomenute stvari, moguće je razvijati ASP.NET Core MVC aplikaciju na Linux operativnom sistemu.

# 4. zaključak

Kao i ostala okruženja, ASP.NET Core MVC ima svoje prednosti i mane od kojih su neke bitnije prikazane i objašnjene u tekstu. Ovaj tekst je obuhvatio samo jedan jako mali deo celog rada sa ASP.NET Core MVC-om. Kada se programer odlučuje za okruženje i jezik u kom će raditi treba da posmatra više aspekata, tako da dati striktan zaključak okruženje je loše ili nije se ne može, posebno ne iz ovako malog sumiranja. Ukratko bitnije stvari iz tekst gore: bi rekao da je prilikom razmišljanja o korišćenju ovog okruženja bitno uzeti u obzir cenu i vreme koje je potrebno da se dobro savlada ovo okruženje, kao i svo potrebno znanje. Naravno, dosta dokumentacije postoji vezano za probleme i rad sa ovim okruženjem. Bitno je uzeti u obzir da koristi MVC model što na neki način forsira da se piše uredan kod. Sa druge strane, MVC gubi svoj smisao u nekim situacijama kada je modularnost izražena na drugi način. Rutiranje je isto jedna bitna i dobra osobina

# **Kreiranje aplikacije**

# Literatura

1. Learning ASP.NET Core MVC Progrmming, Mugilan T.S. Ragupathi
2. Pro ASP.NET Core MVC, Adam Freeman, 6th edition
3. http://ccftp.scu.edu.cn:8090/Download/fb3ea1d2-5dda-4c41-bfde-51d08e08a3e1.pdf
4. https://docs.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-3.1
5. [https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller#Goals\_of\_MVC](https://en.wikipedia.org/wiki/Model–view–controller" \l "Goals_of_MVC)
6. https://en.wikipedia.org/wiki/Separation\_of\_concerns
7. https://www.tutorialspoint.com/asp.net\_core/asp.net\_core\_middleware.htm
8. [https://code-maze.com/routing-asp-net-core-mvc/#attributerouting](https://code-maze.com/routing-asp-net-core-mvc/" \l "attributerouting)
9. <https://docs.microsoft.com/en-us/aspnet/core/fundamentals/routing?view=aspnetcore-3.1>
10. https://www.yogihosting.com/aspnet-core-url-routing/
11. <https://kb.iu.edu/d/adnz>
12. https://exceptionnotfound.net/attribute-routing-vs-convention-routing/
13. http://softwaretestingfundamentals.com/unit-testing/
14. https://www.guru99.com/unit-testing-guide.html
15. https://docs.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/test-asp-net-core-mvc-apps
16. https://www.codingame.com/playgrounds/35462/creating-web-api-in-asp-net-core-2-0/part-2---unit-tests
17. https://www.nuget.org/packages/Microsoft.NETCore.Platforms/
18. https://subscription.packtpub.com/book/application\_development/9781785886751/1/ch01lvl1sec15/creating-an-asp-net-core-mvc-application-on-linux
19. https://docs.microsoft.com/en-us/aspnet/core/host-and-deploy/linux-apache?view=aspnetcore-3.1
20. https://blog.todotnet.com/2017/07/publishing-and-running-your-asp-net-core-project-on-linux/

1. Jasno mi je sta su design patterns, kako i zasto se koriste, ali ih znam samo pod tim nazivom, ne znam da prevedem ...ne treba ni da prevodiš može samo italic... sve što zadržavaš na engleskom stavi italic [↑](#footnote-ref-2)
2. Tekst se oslanja na ASP.NET 5 verziju [↑](#footnote-ref-3)
3. Pipeline čini niz elemenata koji obrađuju podatke povezani u neki niz, izlaz jednog elementa je ulaz drugog [↑](#footnote-ref-4)