

UNIVERZITET U BEOGRADU

MATEMATIČKI FAKULTET

SEMINARSKI RAD

**Primena XML Tehnologija u .NET-u**

Predmet: XML tehnologije

Nastavnik: prof. dr Dušan Tošić

Student: Miloš Stanković, 2014/2013

Beograd, 2013/2014

SADRŽAJ

[MVC obrazac dizajniranja - 2 -](#_Toc389325581)

# MVC obrazac dizajniranja

Osnovna ideja obrazaca dizajniranja (*design patterns*) je da izdvoje nacrt rešenja za situacije (probleme) koje se često javljaju i ponavljaju u različitim sferama tokom razvoja softvera. Obrazac dizajna ne predstavlja gotovo rešenje problema koje se može direktno pretočiti u kod. To je opis rešenja problema koje se može primeniti u različitim situacijama.

Obrasci dizajniranje nisu vezani ni za jedan programski jezik, biblioteku (*framework*) i slično, već predstavljaju rešenja koja su se kroz praksu dokazala kao dobra (najčešće i najbolja). Zbog toga, oni mogu da značajno ubrzaju razvoj softvera. Nema potrebe trošiti vreme na pronalaženje rešenja problema, već se može odmah krenuti sa implementacijom obrasca dizajniranja za konkretni problem. Druga veoma bitna prednost obrazaca dizajniranja je ta što poboljšavaju komunikaciju između programera. Programeri mogu da lakše stvore sliku o nekom softverskom proizvodu ukoliko znaju koji od obrazaca dizajniranja je bio korišćen tokom njegovog razvoja.

MVC (*Model-View-Controller*) predstavlja primer jednog obrasca dizajniranja. On nije povezan konkretno sa AngularJS-om, već je uspešno implementiran u različitim programskim jezicima. MVC predstavlja arhitekturu na kojoj se zasniva najveći broj modernih veb, Android, iOS i desktop aplikacija. Na slici x prikazan je MVC obrazac dizajniranja koji će u nastavku biti detaljno opisan.



MVC se sastoji od tri osnovne logičke komponente: model podataka (*model*), prezentacioni deo ili pogled (view), i kontrolni deo (*controller*). Svaka od navedenih komponenti se sastoji od većeg broja klasa.

Model se sastoji od čistih podataka koji se dobijaju iz baze podataka, veb servisa ili iz nekog drugog izvora, i koje je potrebno prikazati korisniku. Zbog toga su modeli najčešće POJO (*Plain Old Java Object*) objekti [x]. POJO klase su jednostavne klase koje ne nasleđuju neke specijalne Java klase. One se sastoje od skupa (privatnih) polja i odgovarajućih *get* i *set* metoda, a njihova uloga je u prenosu podataka.

Pogled je komponenta koju korisnik direktno vidi. Pogled prikazuje podatke iz modela, pri čemu ne zna ništa o semantici tih podataka. U eri mobilnih uređaja i velikog broja različitih ekrana na kojima se pogled prikazuje, česta je pojava da se za određeni model definiše nekoliko pogleda.

Kontroler predstavlja deo MVC-a koji koordinira između modela i različitih pogleda na aplikaciju. On osluškuje događaje koji se dešavaju na pogledu (npr. klik na dugme) i izvršava odgovarajuće akcije pridružene tim događajima. Neke od tih akcija mogu se odnositi na dohvatanje novih podataka iz baze podataka od kojih se formira novi model, a zatim se taj model prosleđuje nekom drugom pogledu koji korisniku prikazuje zahtevane podatke.

Prva očigledna prednost MVC obrasca jeste razdvajanje prezentacionog dela aplikacije od njenog logičkog dela. Podrška različitim vrstama korisnika koji koriste različite vrste uređaja je standardni zahtev današnjih veb aplikacija. Korisnički interfejs mora biti različiti ako zahtev za aplikacijom stiže od desktop računara ili mobilnog uređaja. U oba slučaja model je isti, tj. predstavlja iste podatke koje je potrebno prikazati. Kontroler ima ključnu ulogu u izboru pogleda koji će prikazati zahtevane podatke. Pored izdvajanja pogleda iz poslovne logike aplikacije, MVC smanjuje kompleksnost velikih aplikacija. Kod je struktuiraniji, pa je samim tim jednostavnije upravljati njime, testirati ga i ponovo koristiti.

# MVC i AngularJS

## Javascript MVC biblioteke

Moderne veb aplikacije se sastoje od skupa kontrolera, modela i pogleda. Obično postoji jedan osnovni kontroler koji prihvata sve zahteve, a zatim ih prosleđuje odgovarajućim specifičnim kontrolerima. Kontoler najpre poziva servisni kod koji izračunava model, a zatim određuje odgovarajući pogled. Postoje čitave biblioteke koje implemeniraju MVC obrazac i omogućuju kreiranje MVC veb aplikacija. Neke od njih su Spring MVC i ASP.NET MVC. Na primer, kod Spring MVC biblioteke kontroleri su posebne Java klase, modeli su (najčešće) POJO Java klase, dok su pogledi JSP (*Java Server Page*) dokumenti koji se prevode u HTML.

Popularnost Javascript-a je dovela do toga da se sve više poslovne logike veb aplikacija odvija na klijentskoj strani. Takođe, sa razvojem AJAX (*Asynchronous Javascript And XML*) tehnologije nastala je ideja o kreiranju jednostranih veb aplikacija (*single page web applications*). Jednostrane veb aplikacije se sastoje od jedne osnovne strane. Tokom interakcije korisnika sa aplikacijom, umesto da se ponovo učitavaju kompletne stranice, AJAX tehnologijom se delovi početne strane zamenjuju drugim delovima, tako da korisnik ima osećaj da radi sa klasičnom desktop aplikacijom. Zbog svega navedenog, količina Javascript koda kod prosečne veb aplikacije u stalnom je porastu, pa se javila potreba za boljom organizacijom skript koda. Tako nastale čitave Javascript biblioteke (frameworks) koje implementiraju različite obrasce dizajniranja, pa i MVC. Najpoznatije biblioteke koje implementiraju MVC obrazac dizajniranja su Backbone.js, Ember.js i AngularJS.

## AngularJS biblioteka

AngularJS je jedna od najpopularnijih Javascript MVC biblioteka. Prvu verziju ove biblioteke objavili su Miško Hevery i Adam Abrons iz kompanije Brat Tech LLC 2009. godine. Ubrzo je Adam Abrons napustio projekat, dok je Miško Hevery nastavio da radi na AngularJS-u u okviru Google-a.

Od trenutka kada se Google uključio u razvoj AngularJS biblioteke, njena popularnost stalno raste, što se može videti na slici x. Prikazana slika predstavlja grafik interesovanja za termine '*angularjs'*, '*backbone.js*' i '*ember.js*' u periodu od januara 2011. do juna 2014. godine. Interesovanje navedenih termina se zasniva na njihovoj učestalosti u pretraživanjima na Google internet pretraživaču. Grafik je dobijen na sajtu Google Trends ([www.google.com/trends](http://www.google.com/trends)), a broj pojavljivanja pomenutih termina skaliran je od 0 do 100.



Pošto se ovaj rad bavi klijentskim (skript) delom veb aplikacije, zbog jednostavnosti u radu će se za njega koristiti termin 'veb aplikacija' ili samo 'aplikacija'.

Pre svega, AngularJS je biblioteka koja definiše različite koncepte za pravilno organizovanje (klijentskog dela) veb aplikacije. Aplikacija se organizuje kao skup modula koji mogu da zavise jedni od drugih, a međusobno referenciranje modula ostvaruje se pomoću tzv. umetanja zavisnosti (*Dependency Injection* - DI). Više o DI može se naći na [x]. Zahvaljujući DI često korišćeni delovi aplikacije mogu se organizovati kao posebni servisni moduli koji se po potrebi mogu umetati i pozivati bilo gde u aplikaciji.

Angular unapređuje HTML posebnim jednolinijskim izrazima (*inline expressions*). U cilju kreiranja efikasnijih pogleda, Angular omogućuje izvršavanje jednolinijskih izraza unutar HTML koda koji se zadaju unutar dvostrukih vitičastih zagrada. U tim izrazima dostupan je Javascript kod pomoću koga je moguće izvršiti određena izračunavanja i prikazati željeni rezultat. Na slici [x] prikazan je jedan jednolinijski izraz i njegov rezultat.



Takođe, Angular proširuje HTML novim elementima i atributima koji se nazivaju direktive (*directives*). Direktive predstavljaju jedno od najmoćnijih funkcionalnosti Angulara. One služe za dodatno struktuiranje HTML koda prilagođavajući ga za podatke u modelu koje je potrebno prikazati. Neke od već definisanih direktiva su *ngApp*, *ngView*, *ngController*, *ngRepeat*, a moguće je i definisati sopstvene direktive. Više o direktivama u nastavku.

Značajna osobina Angular-a je i dvosmerno povezivanje podataka (*two way binding*). Dvosmerno povezivanje podrazumeva menjanje modela paralelno sa interakcijom korisnika. Na primer, kada korisnik unosi neki tekst u tekstualno polje HTML forme, paralelno sa unosom menja se i modelni objekat. To znači da nije potrebno osluškivati specifične događaje, reagovati na njih i eventualno ručno menjati HTML, već takve akcije Angular sam uspešno obavlja.

Zahvaljujući DI, Angular omogućuje jednostavno i efikasno testiranje Javascript koda. Postoje posebne biblioteke koje služe za testiranje AngularJS aplikacije. U ovom radu testiranje aplikacije neće biti razmatrano, a više o testiranju može se naći na [x].

## Osnovni primer AngularJS aplikacije

U ovoj sekciji biće prikazana najjednostavnija AngularJS aplikacija. U ovom i u narednim primerima koristiće se verzija 1.2.17 AngularJS biblioteke. Na slici [x] prikazana je 'Zdravo svete' aplikacija, gde je na levoj strani slike dat kod aplikacije, dok je na desnoj strani prikazan rezultat izvršavanja aplikacije.

