Pravljenje MVC aplikacije kojom će se prikazati podaci iz tabele

.

U složenim aplikacijama koje prikazuju korisniku ogromne količine podataka, u nekim slučajevima je bolje razdvojiti kod koji se bavi podacima od onog koji se bavi interfejsom, tako da razvoj oba postane lakši i jednostavniji.

Model-View-Controller (MVC) rešava ovaj problem razdvajanjem podataka i biznis logike od njihovog prikaza i interakcije sa korisnikom, uz to uvodeći i komponentu zaduženu za koordinisanje prve dve.

Ukoliko bi smo želeli da napravimo MVC aplikaciju kojom bi prikazivali podatke iz tabele, onda je potrebno napraviti neke promene u projektu

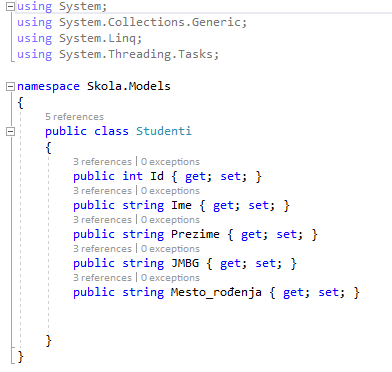
Najpre moramo konfigurisati usluge aplikacije tako što u klasu *Startup.cs* unesemo *services.AddMvc ()*, kojeg poziva Web Host, a kojim činimo da MVC usluge budu dostupne sistemu. Naravno ovaj servis je upotrebljiv samo ako se prethodno doda *app.UseMvc ()*, odnosno najavi aplikaciji da MVC učestvuje u fazi obrade zahteva.



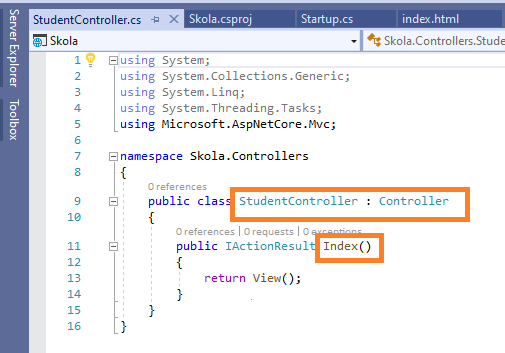
U samom kodu su dodate još dve metode *app.UseDeveloperExceptionPage ()* i *app.UseHsts ().* Metoda *UseDeveloperExceptionPage ()* je korisna u fazi razvoja, jer njenim pozivanjem dobićemo obaveštenje o eventualnoj greški u kodu koja je postavljena u pregledaču. Najbolje je metodu *UseDeveloperExceptionPage ()* smestiti unutar bloka **if** i da se poziva samo ako razvojna metoda *IsDevelopment ()* vraća **true** u suprotnom se izvršava neki drugi kod za rukovanje greškama, odnosno u našem slučaju se poziva metoda *app.UseHsts ()* koji nam pomaže da zaštitimo web lokaciju od slanja kolačića.

Posle sređivanja klase *Startup.cs* u projekat kreiramo folder *Models* i u njemu kreiramo novu klasu *Studenti.cs.* Model objekat poznaje sve podatke koje treba prikazati. Takođe može da definiše operacije koje manipulišu ovim enkapsuliranim podacima. Kreiranje novog modela se odvija tako što desnim klikom na folder Models izaberemo opcije Add > Class…

Kao rezultat dodavanja klase, formiran je novi fajl i automatski se otvara editor za uređivanje koda klase. U mom primeru klasa *Studenti* predstavljaće stavke u bazi. Takođe svaki objekat *Studenti* predstavlja i jedan red u tabeli, a svako svojstvo klase predstavlja kolonu u tabeli. Dodavanjem svojstva ovoj klasi formira se opis kolona buduće tabele.



Po sređivanja klase *Studenti* u projekat kreiramo folder *Controllers* i u njemu kreiramo novu klasu kontroler (*StudentController.cs*), koji je odgovorni za tok aplikacije i rad sa unetim podacima i podacima koji se prikazuju.

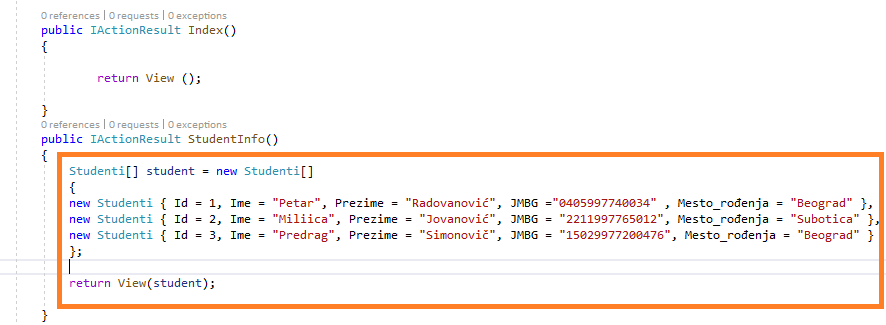


Kao što se vidi, novi kontroler je kreiran kao klasa izvedena iz klase *Controller*. Takođe, automatski je dodata jedna metoda po njegovom kreiranju. Ta metoda je *Index* i ona predstavlja podrazumevanu metodu kontrolora koja će se koristiti ako se eksplicitno ne navede metoda.

Na kraju kreiramo novi folder *Views* (pogledi) sa podfolderom *Student.* Desnim klikom na podfolder *Student*  iz padajućeg menija izaberemo Add >View... i dodamo novi pogled *index.cshtml,* koji ćemo povezati sa kontrolerom *StudentController.cs*.



Pogledi su odgovorni za definisanje prikaza i njima upravljaju kontroleri. Poglede nije moguće direktno prikazati u web čitaču tj. njihov prikaz se uvek generiše od strane kontrolera.

 Po kreiranju ova tri foldera u klasi *StudentController.cs* ćemo kreirati još jednu metodu S*tudentInfo.cs*. Ona će sadržati promenljivu tipa Studenti[] student = new Studenti[] {} i pomoću nje ćemo popunjavati tabelu nizom objekata koji su u njoj definisani.

Na kraju je potrebno u klasi *Starup.cs* definisati rutu kojom se URL mapira na metode kontrolera. U našem primeru su definisani podrazumevani kontroler (Student) i metoda (Index) koji odgovaraju putanji View fajla, odnosno putanju do index strane za prikaz *Views/Student/index.cshtml*.

