KREACIJSKI PATTERNI

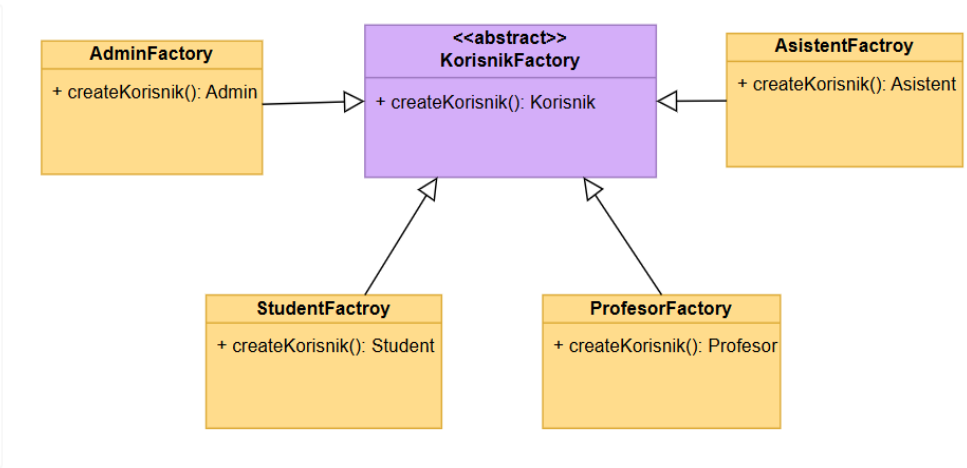
- Uvod

U cilju poboljšanja fleksibilnosti, održavanja i proširivosti sistema, uvođenje kreacijskih paterna u postojeću arhitekturu predstavlja značajan korak ka kvalitetnijem razvoju softverskog rješenja. U ovom dokumentu predstavljamo potrebu i primjenu dva kreacijska paterna – Singleton i Factory Method – u okviru sistema koji koristi MVC arhitekturu, kao I Prototype, Abstract Factory i Builder.

- Factory Method

On omogućava kreiranje objekata bez direktnog pozivanja njihovih konstruktora, već kroz apstraktne metode koje se implementiraju u podklasama.

Primjena u sistemu:
Kod kreiranja različitih tipova korisnika (Student, Profesor, Asistent, Admin), Factory Method se može koristiti da bi se izbjegla direktna instanciranja unutar kontrolera. Također, može se koristiti za kreiranje različitih tipova Pitanje, zavisno od kategorije (AI, EE, TK, RI).

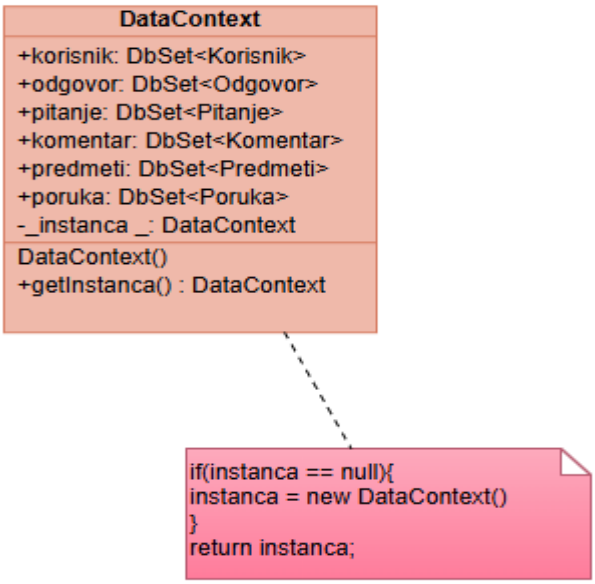


Zatim u kontroleru koristimo ovakvu kreaciju.

- Singleton pattern

Osigurava da postoji samo jedna instanca određene klase i omogućava globalni pristup toj instanci.

Primjena u sistemu:
U našem sistemu, klasa DataContext predstavlja centralizovano mjesto za pristup podacima. Kako bi se izbjegle višestruke instance i potencijalni problemi sa dosljednošću podataka, potrebno je ovu klasu pretvoriti u Singleton.



- **Prototype Pattern**

Prototype pattern omogućava kreiranje novih objekata kopiranjem već postojećih objekata (šablona), što štedi vrijeme i smanjuje potrebu za ponovnim unosom podataka.

Moguća primjena u sistemu:
Zamislimo situaciju gdje korisnik želi unijeti više vrlo sličnih pitanja (npr. iz istog predmeta, sa istim kategorijama, ali različitim tekstom pitanja). Umjesto da svaki put kreira novo pitanje od nule, može iskoristiti postojeće pitanje kao šablon i samo izmijeniti tekst.

Gdje bi se koristilo:
U PitanjeController, može se dodati metoda ClonePitanje(int pitanjeId) koja vraća kopiju postojećeg pitanja za dalju izmjenu.

- **Abstract Factory Pattern**

Abstract Factory pattern omogućava kreiranje čitavih porodica povezanih objekata koji međusobno sarađuju, bez navođenja njihovih konkretnih klasa.

Moguća primjena u sistemu:
U sistemu koji se može proširiti za različite univerzitete, institute ili vrste obrazovnih ustanova, svaki od njih može imati vlastitu verziju korisnika (npr. drugačije uloge ili prava), predmeta, pitanja i poruka.
Abstract Factory omogućava da se sve komponente za određeni kontekst (npr. "Privatni fakultet", "Državni univerzitet") kreiraju zajedno.

Gdje bi se koristilo:
Umjesto da se pojedinačno kreiraju Korisnik, Predmet, Pitanje, može se napraviti npr. UniverzitetFactory interfejs koji kreira sve te komponente. Imali bismo konkretne fabrike: DrzavniUniverzitetFactory, PrivatniUniverzitetFactory.

- **Builder Pattern**

Builder pattern omogućava kreiranje složenih objekata korak po korak. Koristan je kada postoji mnogo opcionalnih parametara ili kada kreacija zahtijeva više faza.

Moguća primjena u sistemu:
Zamislimo da korisnik dodaje novo pitanje (Pitanje), a zatim postepeno dodaje odgovore (Odgovor), komentare, kategorije i druge detalje. Umjesto da sve to unosi odjednom, Builder pattern omogućava postepeno dodavanje dijelova.