Fakultet elektrotehnike i računarstva

Zavod za primijenjeno računarstvo

**Objektno oblikovanje**

Laboratorij profila

*Ratnici s Marsa*

Zvonimir Gračak 0036473216

Filip Jurčić 0036475588

Pavao Križić 0036470053

Ante Šućur 0036474806

Zagreb, 04.02.2017.



*League Assist Manager*

*OR mapiranje*

Sadržaj

[1. Uvod 1](#_Toc474006970)

[2. Implementacija 2](#_Toc474006971)

[2.1. Konfiguracija 2](#_Toc474006972)

[2.2. Definicija modela 2](#_Toc474006973)

[2.3. Definicija mapiranja 3](#_Toc474006974)

[2.4. Kreiranje session objekta 4](#_Toc474006975)

[2.5. Rad s objektima (repository) 4](#_Toc474006976)

[3. Zaključak 7](#_Toc474006977)

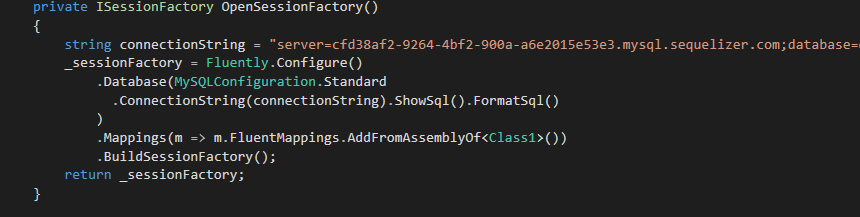
# Uvod

U našem seminarskom radu, kako bi omogućili automatsku i transparentnu pohranu objekata u tablice relacijske baze podataka, korišteno je OR (Object/relational) mapiranje. Kod OR mapiranja se u principu radi o transformaciji iz jedne reprezentacije u drugu, gdje su velike prednosti veća produktivnost, jeftinije i lakše održavanje te nezavisnost o samoj tehnologiji SUBP-a. Konkretnije, u seminarskom radu, za OR mapiranje, koristili smo *Fluent NHibernate* iz razloga što, umjesto pisanja SQL class library code-a za rad s bazom podataka, ima jednostavno sučelje za CRUD (create/read/update/delete) operacije nad objektima te razredima i njihovim svojstvima. Kod *Fluent NHibernate-a* potrebno je definirati model, postaviti konfiguraciju parametara za spajanje između same aplikacije i baze podataka, definirati mapiranje izgrađenog modela, napraviti *session* te je nakon toga moguće raditi s objektima. Ono što je najvažnije u svemu tome, je to što treba odrediti način na koji će svaki od objekata biti pohranjen u njegovu odgovarajuću tablicu u bazi podataka, odnosno treba ispravno definirati mapiranje objekata. NHibernate nudi tri vrste načina mapiranja objektnog modela u relacijski, a to su *table per concrete class, table per hierarchy class* i *table per* subclass. U našoj aplikaciji od početka je realizirano *table per concrete class* mapiranje, nakon čega smo tek na kraju samog projekta došli do zaključka da je za naš model itekako bilo potrebno raditi *table per subclass* mapiranje. U nastavku ovog dokumenta, opisati ćemo kako je izvedena konkretna implementacija *Fluent NHibernate-a* na našoj aplikaciji.

# Implementacija

Za potrebe OR mapiranja koristili smo Fluent NHibernate za razliku od običnog NHibernatea koji koristi XML za konfiguriranje i mapiranje klasa. U nastavku ćemo opisati korištenje NHibernatea u pet koraka na primjeru jedne klase.

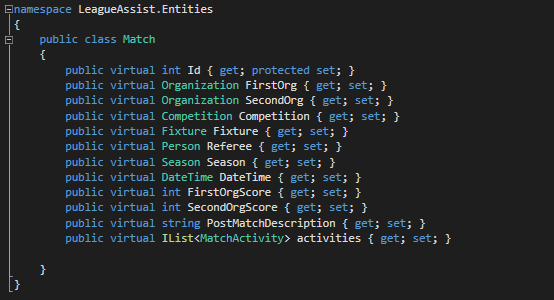
## Konfiguracija



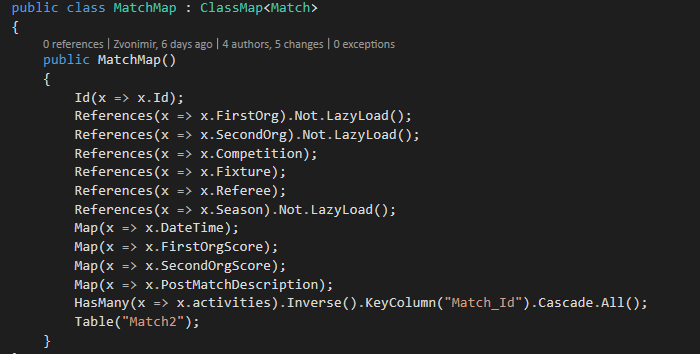
Na slici vidimo isječak koda iz naše klase koja kreira session factory te služi za otvaranje sessiona. Prilikom konfiguracije session factorya naveli smo tip baze koju koristimo, connection string za spajanje na bazu, dodatne parametre (npr. ShowSql i FormatSql koji nam omogućuju ispis SQL naredbi koje NHibernate kreira i provodi nad bazom) te naveli gdje da se gleda mapiranje klasa.

## Definicija modela

Kako bi definirali model naše domene, za svaku tablicu i view koji su bili kreirani u našoj bazi smo napravili klasu istog imena kao i tablica (odnosno view). Također i atributi u klasama su istog imena kao i atributi u tablicama iz baze.



## Definicija mapiranja



Prilikom mapiranja klasa koristili smo posebne klase koje kao baznu klasu imaju klasu definiranu u Fluent NHibernateu ClassMap. Tamo smo popisali mapiranje za sve atribute pojedine klase kao što se vidi na slikama. Oznaka Id označava da je ta kolona odnosno atribut identifikator klase u bazi. References označava strani ključ na neki drugi razred odnosno tip objekta. Map označava običan atribut pojedine klase. Oznaka HasMany označava master detail odnos između dvije klase odnosno tablice u bazi i olakšava nam spremanje podataka u bazu na način ako idemo spremiti master podatak uz njegove detail podatke NHibernate će ih sam spremiti automatski u obje tablice, svaku u svoju.

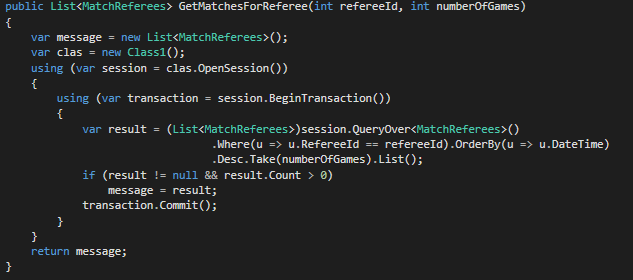
## Kreiranje session objekta

Tijekom inicijalizacije aplikacije kreira se jedna istanca SessionFactorya kako bi se mogao ostvariti pristup bazi podataka te kako bi mogli pristupiti podatcima o mapiranju. Metodu OpenSession() zato koristimo u repozitorijima jer preko njih naše aplikacije pristupaju podatcima u bazi, te su zadužene za ažuriranje i spremanje podataka.

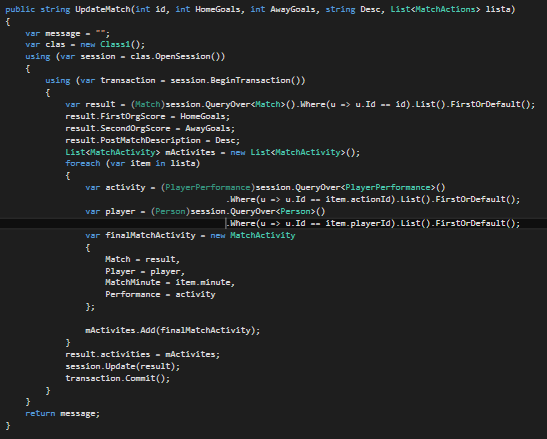
## Rad s objektima (repository)

Repozitorije smo organizirali na način da imamo repozitorij za svaku važniju komponentu u sustavu. U repozitoriju se nalaze metode koje služe za dohvat i spremanje podataka u bazu odnosno služe za interakciju s bazom podataka. Tu su uz osnovne operacije definirani i malo složeniji upiti na bazu. Svaki repozitorij je definiran sučeljem koje govori koje sve metode mora repozitorij sadržavati kako bi zadovoljio sučelje. Svaka od metoda koristi svoju transakciju za spremanje i dohvat podataka. Nakon obavljanja upita transakcija se commita.

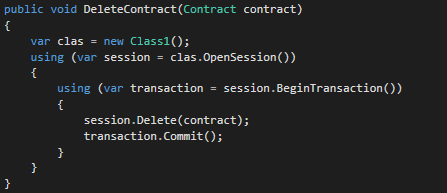
Najosnovniji način za dohvat objekta iz baze je uz pomoć funkcije Get<Tip objekta>(id) koja dohvaća objekt preko njegovom ID-a.



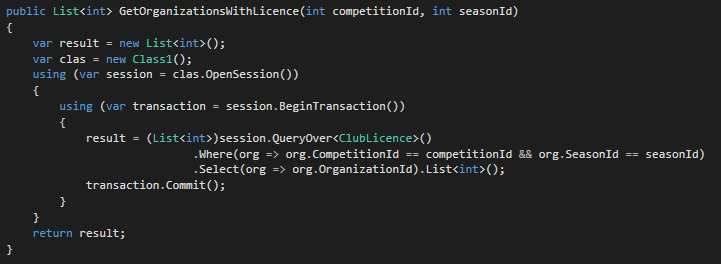
Najčešće smo podatke spremali uz pomoć SaveOrUpdate(tip objekta) funkcije koja govori NHibernate-u da ukoliko objekt ima ID potraži u bazi i izmijeni ga, u protivnom unese novi objekt u bazu podataka.



Za brisanje podataka koristili smo Delete(objekt) funkciju koja NHibernateu šalje objekt koji želimo izbrisati iz baze. Valja napomenuti kako pri brisanju podataka moramo paziti na redoslijed objekata koje brišemo, jer moramo paziti da ne pokušavamo izbrisati objekt koji se u nekoj drugoj tablici referencira.



Za dohvat objekata ili liste objekata koristili smo LINQ upite koje NHibernate podržava.



Primjer upita nad tablicom OrgCompetition koja dohvaća objekte koji imaju ID natjecanja i sezone jednake zadanima. Prilikom dohvaćanja uzimaju se samo ID-evi organizacija koje zadovoljavaju te uvijete odnosno kombinira se SELECT i WHERE dio SQL naredbe kako bi se dobila lista integera.

# Zaključak

Po uzoru na gore opisani primjer mapiranja jedne tablice iz baze smo napravili entitete, map klase, repozitorije te procesore preko kojih pozivamo metode za rad s bazom (spremanje, ažuriranje i brisanje podataka). Mozemo zakljuciti da je rad s Fluent NHibernateom puno jednostavniji nego npr. s NHibernateom pisanog putem XML-a koji zahtjeva puno veći kod, te poznavanje jezika, dok Fluent NHibernate možemo pisati u C# klasama te imamo manji kod. Također valja napomenuti kako je Nhibernate „strongly typed“ te promjenom imena atributa ili naziva klase u odnosi na atribute i imena tablica u bazi će rezultirati greškom, dok NHibernate pisan putem XML-a neće reagirati na bilokakve promjene.