**Uvod u Entity Framework**

**00:11:49**

Jedinica: 6 od 19

**+Rezime**

Baza podataka modeluje pojmove iz realnog sveta korišćenjem skupa tabela, indeksa, pogleda i drugih elemenata. Tako se može reći da je baza podataka apstrakcija informacija koje postoje u realnom svetu. Ipak, aplikacija ne može direktno koristiti podatke iz baze podataka zato što je ona sposobna da podacima rukuje u nešto drugačijem obliku od onog koji postoji u relacionim bazama podataka. Stoga je za korišćenje podataka iz baze unutar aplikacije, neophodno stvoriti još jedan nivo apstrakcije nad realnim podacima. Upravo takav, dodatni nivo apstrakcije, u prethodnim lekcijama kreiran je na nekoliko različitih načina.

U prvom modulu ovoga kursa, apstrahovanje relacionih podataka učinjeno je korišćenjem najprimitivnijih tehnika, koje su podrazumevale ručno kreiranje klasa modela. U drugim modulu, za kreiranje klasa modela iskorišćena je pomoć razvojnog okruženja, tako da su klase modela bile generisane u procesu koji se naziva objektno-relaciono mapiranje.

U nastavku kursa, za vršenje procesa objektno-relacionog mapiranja biće korišćena tehnologija Entity Framework.

**Šta je Entity Framework?**

ADO.NET Entity Framework je ultimativna Microsoft tehnologija za razvoj aplikacija koje rukuju podacima. Entity Framework u kombinaciji sa razvojnim okruženjem Visual Studio obezbeđuje sveobuhvatni ekosistem koji omogućava razvoj širokog spektra aplikacija, uključujući desktop, web, [cloud](https://www.link-elearning.com/linkdl/opisPojma.php?id=142484) i service aplikacije.

Entity Framework primarno je alat za vršenje objektno-relacionog mapiranja, koji automatski obezbeđuje nešto ranije pomenuti drugi nivo apstrakcije nad podacima. Tako EF pojednostavljuje pristup podacima, oslobađajući programera potrebe za direktnim obraćanjem bazi podataka. Za razliku od toga, podacima je moguće pristupiti pisanjem upita nad strogo tipiziranim klasama, a EF će u pozadini obaviti svu neophodnu interakciju sa skladištem.

U svom nazivu Entity Framework sadrži jednu ključnu reč, bez čije definicije nastavak upoznavanja sa ovom tehnologijom ne bi imao smisla. Reč je o entitetu.

**Šta je entitet?**

Iz perspektive aplikacije koja koristi podatke, entitet je pojam koji se odnosni na podatke povezane sa određenim objektom aplikacije. Na primer, u prethodim lekcijama kreirana je klasa sa nazivom *Employee*, koja je modelovala pojam zaposlenog. Takva klasa je imala svojstva koja su bliže određivala pojam zaposlenog. Sa stanovišta objektno-relacionog programiranja, ovako modelovan pojam je kompletan zato što enkapsulira sve svoje osobine, pa se objekti takve klase mogu nazvati entitetima.

Ipak, u bazi podataka, podaci o jednom zaposlenom mogu biti rasuti preko više povezanih tabela. Tako bi u bazi podataka entitet bio jedan red u tabeli, ali sa informacijama iz redova svih povezanih tabela, koji bi određivao pojam jednog zaposlenog.

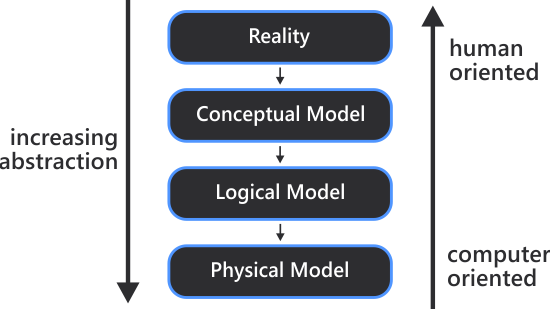
Da bi se sve ovo na pravi način razumelo, neophodno je pažnju posvetiti načinima na koji se podaci mogu modelovati.

**Različiti pogledi na podatke**

Generalno, postoje tri različita načina na koje je moguće posmatrati podatke. Takvi različiti pogledi se mogu nazvati i tipovi modela.

* + **Fizički** model podataka je u potpunosti zavisan od tehnologije koja se koristi za perzistenciju. Takva tehnologija može biti relaciona baza podataka, XML dokument ili [NoSQL](https://www.link-elearning.com/linkdl/opisPojma.php?id=142485) skladište. Ovakav model mora ispoštovati konkretna sintaksna pravila skladišta koje koristi. Svi elementi takvog modela su optimizovani kako bi se sistemu za upravljanje bazama ([DBMS](https://www.link-elearning.com/linkdl/opisPojma.php?id=142486)) olakšala manipulacija i čuvanje podataka. Tako je fizički model za čuvanje podataka efikasan za DBMS, što se ne može reći i za samu aplikaciju, koja je potrebna za rukovanje takvim podacima. Na primer, podaci o jednom zaposlenom se mogu nalaziti u nekoliko različitih, međusobno povezanih tabela. Spajanjem takvih podataka, mogle bi se dobiti sve informacije o jednom zaposlenom.
  + **Logički** model podataka je nezavistan od konkretne tehnologije za perzistenciju i bavi se opisivanjem podataka sa poslovne tačke gledišta. Iako su danas među najkorišćenijim relaciona skladišta, nema nikakvog ograničenja po kome logički model mora biti kreiran korišćenjem relacionih načela. Ipak, ukoliko je tako nešto slučaj, ovakav model definiše relacije između entiteta, kardinalnosti i dozvoljene vrednosti. Podaci često imaju definisanu preciznosti i dužinu, ali nisu strogo vezani ni za jedan sistem za rukovanje podacima.
  + **Konceptualni** model  je logički model baze podataka, preveden u objektni model aplikacije. Ovaj model je najpogodniji za programera aplikacije, jer su podaci već pripremljeni i dostupni kroz objekte. Programer je oslobođen konstrukcije kompleksnih upita i funkcionalnosti za rukovanje bazom i bazi se može obraćati isključivo putem objekata. Konceptualni model je moguće samostalno kreirati, ali simptomatika koja se pojavljuje prilikom mapiranja elemenata logičkog modela u konceptualni, omogućava postojanje automatizovanog sistema, koji većinu posla obavlja sam. Entity Framework je upravo jedan takav sistem, a Entity Data Model (EDM) je konceptualni model koji ovaj sistem generiše.

Slika 6.1 ilustruje tri opisana modela podataka i njihov međusobni odnos.



*Slika 6.1 – Različiti pogledi na podatke*

Prilikom korišćenja bazičnog pristupa za komunikaciju sa bazom podataka programer je taj koji mora da uzme u razmatranje sva tri navedena modela. On mora precizno znati koja tabela čuva koje podatke, ali i to kakvi su međusobni odnosi između tabela. Programer je zadužen i za pisanje koda koji će obaviti povezivanje baze i aplikacije.

Entity Framework olakšava sve opisane poslove, tako što omogućava programeru da se fokusira na rad sa entitetima, a da zanemari fizičku i logičku strukturu podataka. Već je rečeno da se jedan entitet korišćenjem logičkog i fizičkog modela može nalaziti u više različitih povezanih tabela. Ipak, posmatrano sa objektne tačke gledišta, entitet je uvek jedna zaokružena celina koja enkapsulira sve osobine tipa iz realnog sveta koji predstavlja. Tako je rukovanje entitetima za programera mnogo jednostavnije, što je i konačni cilj.

**Tehnike za interakciju sa skladištem**

Postoje tri osnovna načina na koje je moguće koristiti Entity Framework i oni su sledeći:

* + **Database First**. Ukoliko se izgradnja sistema za rukovanje podacima započinje od baze podataka koja već postoji, *Database First* pristup se može koristiti za generisanje konceptulanog modela i mapiranja, odnosno klasa koje reflektuju strukturu baze podataka.
  + **Model First**. Omogućava grafičko kreiranje modela uz pomoć Entity Framework Designera iz koga će potom biti generisana baza podataka.
  + **Code First**. Omogućava definisanje objektnog modela korišćenjem koda. Stoga, ovaj pristup omogućava najveću slobodu prilikom kreiranja modela. Tako definisan sloj modela, može da posluži za generisanje baze podataka.

|  |
| --- |
| ***Napomena:****u nastavku kursa, u modulima o radu sa modelom i rukovanju podacima biće naizmenično korišćeni pristupi Database First i Model First. Pristupu Code First biće posvećen jedan kompletan modul.* |

**Ukoliko je potrebno imati maksimalnu slobodu u kreiranju domenskog modela i samostalno napisati kod klasa koje će poslužiti za instanciranje entiteta, najbolje je koristiti pristup:**

 Database First

 Model First

 Code First

 Nijedan od navedenih

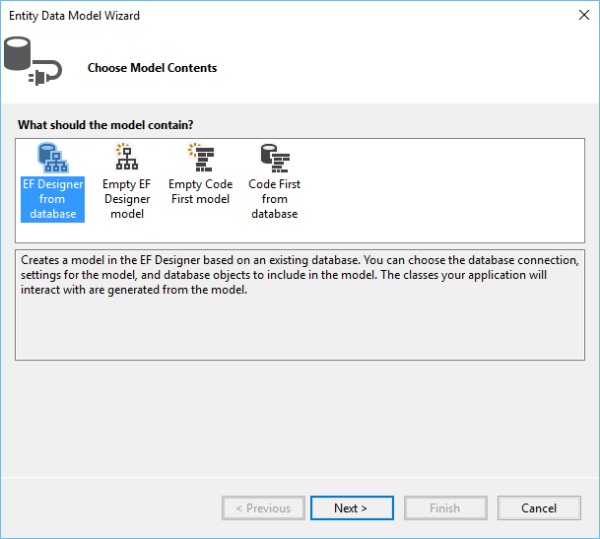
**Korišćenje Entity Frameworka**

Entity Framework funkcioniše u čvrstoj korelaciji sa razvojnim okruženjem Visual Studio. Kako bi se Entity Framework uposlio unutar aplikacije koja se razvija, dovoljno je izvršiti dodavanje stavke na osnovu šablona *ADO.NET Entity Data Model*, kao na slici 6.2.

https://www.link-elearning.com/linkdl/coursefiles/1142/MDA_06_02.jpg

*Slika 6.2 – ADO.NET Entity Data Model šabon*

Postoje četiri opcije na prvoj stranici Entity Data Model čarobnjaka, koje se mogu videti na slici 6.3.

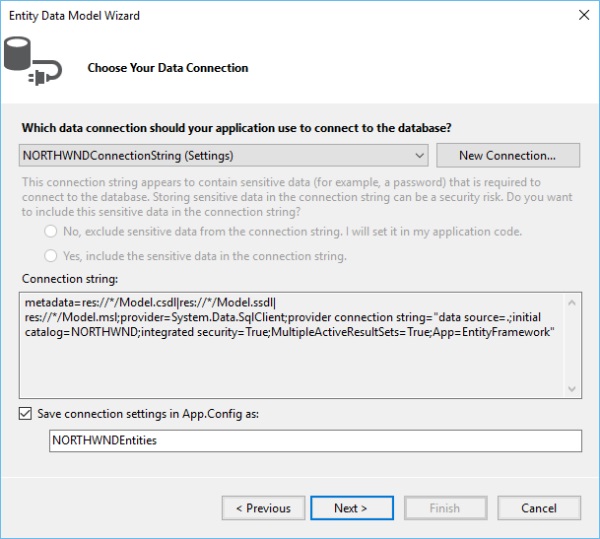


*Slika 6.3 – Različiti načini za korišćenje EF-a*

*EF Designer from database* omogućava generisanje klasa modela na osnovu baze podataka, odnosno *Database First* pristup. *Empty EF Designer model* omogućava *Model First* pristup.

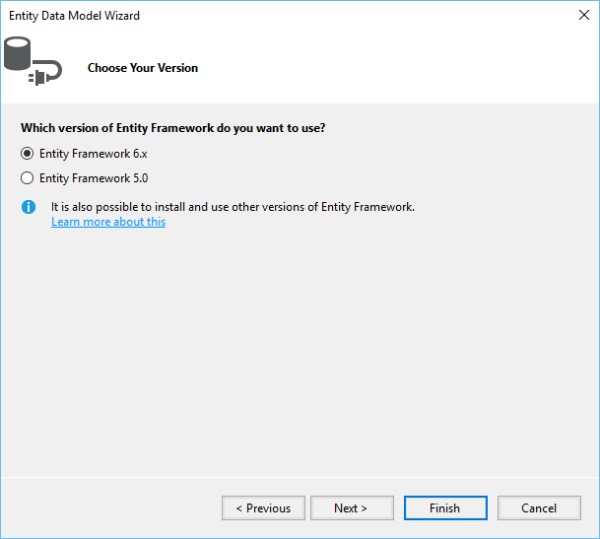
*Empty Code First model* i *Code First from database* omogućavaju primenu *Code First* pristupa. Ovaj primer biće realizovan korišćenjem *EF Designer from database*opcije.

Na sledećoj stranici Entity Data Model čarobnjaka (slika 6.4) potrebno je definisati konekciju sa bazom podataka. Moguće je napraviti novu konekciju odabirom opcije *New Connection...* ili iskoristiti postojeću koja je kreirana u jednoj od prethodnih lekcija. Kao test baza podataka koristiće se Northwind.



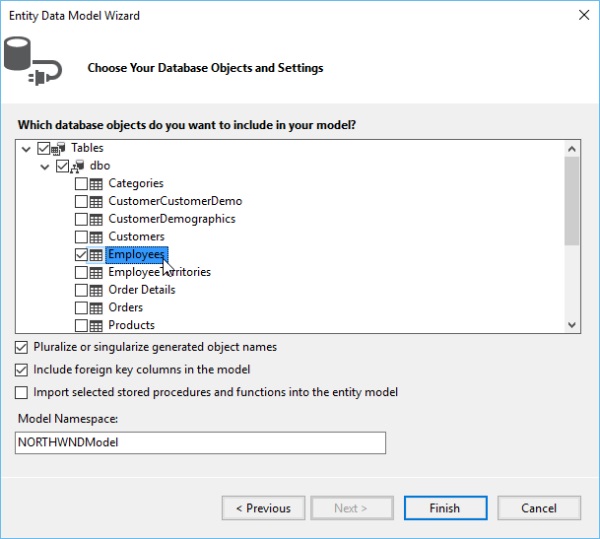
*Slika 6.4 – Kreiranje konekcije*

Sledeći prozor čarobnjaka omogućava odabir verzije Entity Frameworka, koja će biti korišćena, kao na slici 6.5.



*Slika 6.5 – Odabir verzije EF-a*

Poslednji prozor Entity Data Model čarobnjaka omogućava odabir objekata baze podataka koji će ući u model, kao na slici 6.6.



*Slika 6.6 – Odabir objekata baze podataka koji će ući u model*

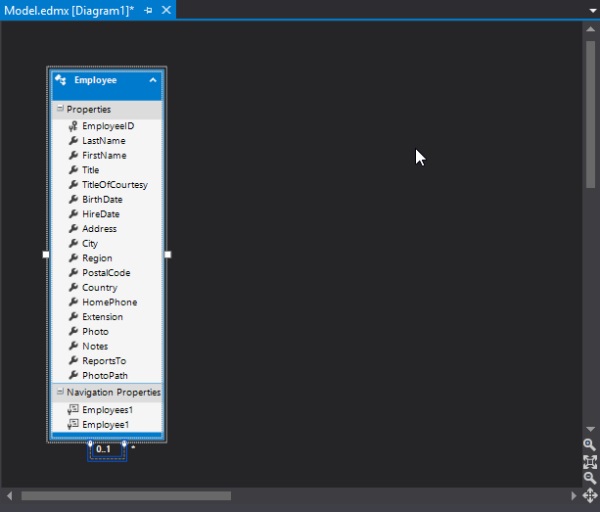
Prilikom odabira objekata baze podataka koji će ući u sastav modela, postoje i tri veoma značajne opcije kojima se može uticati na proces objektno-relacionog mapiranja. Značenje tih opcija opisano je u nastavku.

*Pluralize or singularize generated object names* izvršava singularizaciju imena entiteta, ukoliko su tabele u bazi podataka eventualno u množini. Takođe, čekiranjem ove opcije kolekcije objekata imaće nazive u množini.

Opcija *Include foreign key columns in the model* omogućava uključivanje stranih ključeva u klase modela koje se generišu.

Poslednji CheckBox, *Import selected stored procedures and functions into entity model* vrši automatsko kreiranje Function Imports sekcije za uskladištene procedure i funkcije.

Nakon klika na dugme Finish vrši se kreiranje modela. Radna površina dizajnera za rad sa modelom prikazana je na slici 6.7.



*Slika 6.7 – Entity Data Model (EDM)*

Na ovaj način završeno je kreiranje konceptualnog modela podataka. Podacima se sada može rukovati kao entitetima. Stoga je kreiranje modela jedan od ključnih zadataka, pre nego što se počne sa korišćenjem podataka u aplikaciji. Sledeće lekcija će se baviti upravo time.