**Rukovanje EDM-om**

**00:22:39**

Jedinica: 7 od 19

**Rezime**

* Modelovanje je osnovna funkcionalnosti Entity Frameworka, koja EF razlikuje od prethodnih Microsoft tehnologija za pristup podacima.
* Entity Framework omogućava modelovanje, korišćenjem Entity Data Model Designera.
* Entiteti međusobno mogu graditi nekoliko tipova relacija: One-to-Many, Many-to-Many i One-to-One.
* Tehnika *Entity Splitting* podrazumeva deljanje jednog entiteta u više tabela, odnosno obrnuto.
* Entiteti mogu naslediti osobine drugih entiteta, kao što klase mogu naslediti osobine drugih klasa. Entity Framework poznaje dva tipa nasleđivanja – *Table per Type* i *Table per Hierarchy.*

Ova lekcija biće posvećena osnovnim tehnikama modelovanja pomoću Entity Frameworka. Modelovanje je osnovna funkcionalnost Entity Frameworka, koja EF razlikuje od prethodnih Microsoft tehnologija za pristup podacima. Tako se kod kreira na osnovu modela, a ne na osnovu redova i kolona u relacionoj bazi podataka. EF omogućava modelovanje korišćenjem *Entity Data Model Designer*a.

**Kreiranje Modela**

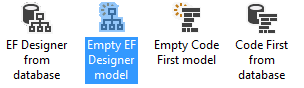
Zamislite situaciju po kojoj je potrebno kreirati aplikaciju za rukovanje osobama. O osobama bi se beležile neke osnovne informacije. U takvoj situaciji bilo bi potrebno modelovati jedan entitet: *Person*.

Za kreiranje novog modela, potrebno je izvršiti desni klik na projekat i odabrati opciju *Add->New Item*. Iz liste dostupnih šablona potrebno je odabrati *ADO.NET Entity Data Model*, kao na slici 7.1.

https://www.link-elearning.com/linkdl/coursefiles/1142/MDA_07_01.jpg

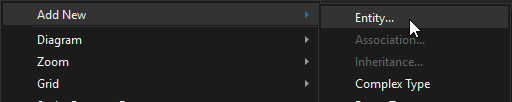
*Slika 7.1 – ADO.NET Entity Data Model šablon*

Na prvoj stranici čarobnjaka sada je potrebno odabrati opciju *Empty EF Designer model*, kao na slici 7.2.



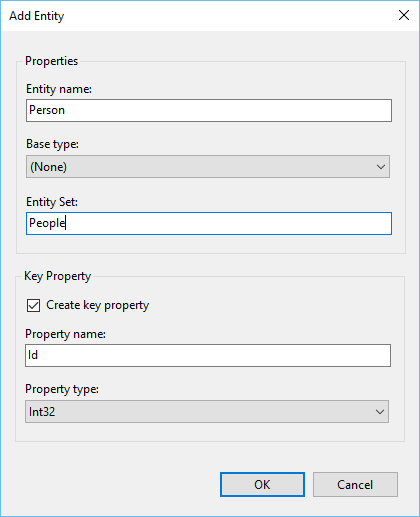
*Slika 7.2 – Empty EF Designer model šablon*

Ovo će rezultovati kreiranjem konceptualnog modela sa praznom radnom površinom za modelovanje. Dodavanje novog entiteta se postiže desnim klikom na radnu površinu i odabirom opcije *Add -> Entity*(slika 7.3.)



*Slika 7.3 – Dodavanje novog entiteta*

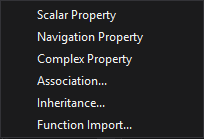
U prozor koji se otvara potrebno je uneti naziv entiteta, kao na slici 7.4.



*Slika 7.4 – Prozor za kreiranje novog entiteta*

Ovako kreiran entitet poseduje samo jedno svojstvo. To je svojstvo Id, koje predstavlja ključ ovog entiteta. Za dodavanje ostalih svojstava, potrebno je izvršiti desni klik na kreirani entitet i odabrati *Add New -> Scalar Property*. Entitetu Person biće dodata svojstva *FirstName*, *LastName* i *DateOfBirth*.

|  |
| --- |
| ***Napomena:****prilikom dodavanja svojstava entitetu može se primetiti da postoji mogućnost dodavanja nekoliko vrsta svojstava, što se može videti na slici 7.5.* |



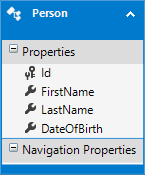
*Slika 7.5 – Prozor za kreiranje novog entiteta*

*Skalarna svojstva su ona čije su vrednosti sadržane u samom entitetu. Takvo svojstvo je na primer FirstName.*

*Navigaciona svojstva se koriste za povezivanje entiteta. Tako ona omogućavaju navigaciju sa jednog entiteta na drugi. Ukoliko postoji entitet Address, svojstvo takvog tipa bi moglo biti navigaciono svojstvo entiteta Person, pa se navigaciona svojstva ponašaju kao strani ključevi u tabelama relacionih baza podataka.*

*Kompleksna svojstva su ona koja omogućavaju skalarnim svojstvima da budu organizovana unutar jednog entiteta.*

Nakon dodavanja svojstava izgled entiteta unutar dizajnera će biti kao na slici 7.6



*Slika 7.6 – Person entitet sa 4 skalarna svosjtva*

Na prikazani način kreiran je jednostavan konceptualni model. Sledeći korak u ovoj lekciji biće generisanje baze podataka.

**Svojstvo sa nazivom LastName najbliže odgovara kojem svojstvu?**

 Skalarnom svojstvu

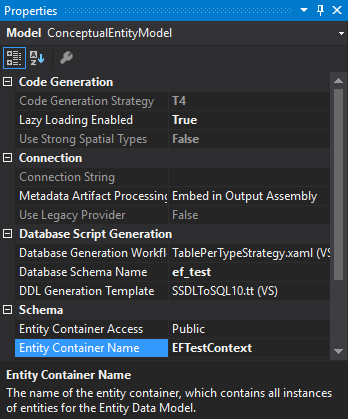
 Navigacionom svosjtvu

 Kompleksnom svosjtvu

**Generisanje baze**

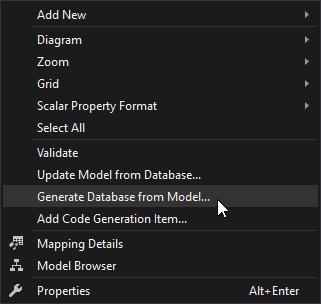
Nakon uspešno kreiranog modela baze podataka na njegovoj osnovi može se izvršiti generisanje skripte koja će kreirani model pretočiti u objekte baze podataka.

Pre pokretanja generisanja, unutar Properties panela je potrebno postaviti naziv šeme koja će se koristiti. Nije zgoreg postaviti i vrednost *Entity Container Name* svojstva, po čemu će se imenovati Data Context klasa. Slika 7.7 dočarava podešavanja.



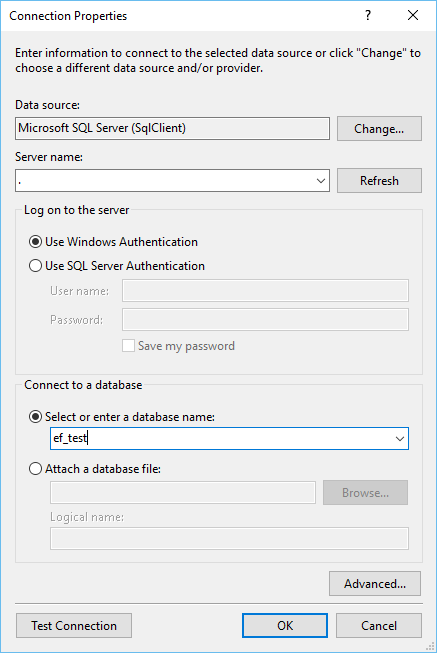
*Slika 7.7 – Podešavanje konceptualnog modela*

Za pokretanje generisanja, dovoljno je odabrati opciju *Generate Database from Model* iz konteksnog menija koji se dobija desnim klikom na radnu površinu dizajnera. Kontekstni meni i pomenuta opcija prikazani su slikom 7.8.



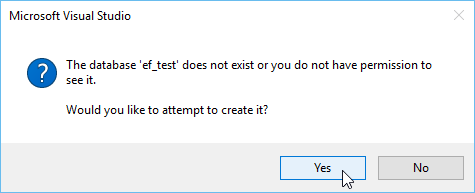
*Slika 7.8 – Opcija za pokretanje generisanja baze podataka na osnovu modela*

Pokretanje prikazane opcije sa slike 7.8 kao rezultat ima otvaranje *Generate Database* čarobnjaka. Prilikom prvog pokušaja generisanja skripte na osnovu modela, čarobnjak će tražiti definisanje detalja konekcije. Drugim rečima, EF-u je potreban konekcioni string kako bi bio u stanju da generiše odgovarajući [DDL](https://www.link-elearning.com/linkdl/opisPojma.php?id=142446). Na taj način je dovoljno odabrati lokaciju servera i uneti željeni naziv baze podataka, kao što vidimo na slici 7.9.



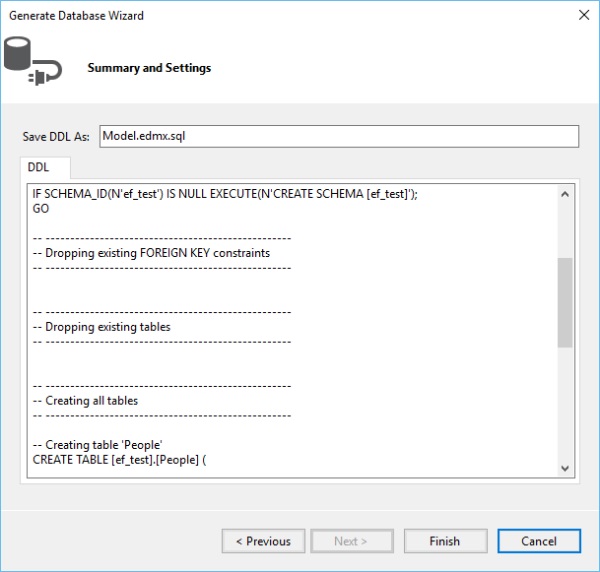
*Slika 7.9 – Podešavanje konekcije*

Ukoliko baza podataka sa unetim nazivom ne postoji, razvojno okruženje će ponuditi da izvrši njeno kreiranje, što se vidi na slici 7.10.



*Slika 7.10 – ADO.NET Entity Data Model šablon*

Poslednji korak Generate Database čarobnjaka prikazuje DDL koji će biti generisan, što se vidi na slici 7.11.



*Slika 7.11 – Generisana skripta za kreiranje baze podataka*

Klikom na dugme *Finish,*u prozoru sa slike 7.11, biva kreiran fajl sa ekstenzijom .sql, koji sadrži skriptu koju je EF generisao. Za kreiranje tabela baze, neophodno je kreiranu skriptu izvršiti na serveru baze.

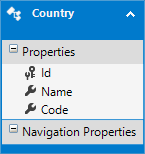
**Modelovanje relacija**

U prethodnom primeru kreirani model sastojao sa iz jednog entiteta. Model se najčešće sastoji iz većeg broja entiteta povezanih određenim relacijama.

|  |
| --- |
| ***Napomena:****entiteti međusobno mogu graditi nekoliko tipova relacija. Ti tipovi su sledeći:*    ***One-to-Many****– najčešći tip relacije, po kojoj red u tabeli A može imati veći broj povezanih redova u tabeli B, ali red u tabeli B može imati samo jedan povezan red u tabeli A. Idealan primer ove relacije grade entiteti Person i Country, ukoliko pretpostavimo da jedna osoba može imati jednu državu stanovanja. U jednoj državi može živeti više osoba, a jedna osoba može živeti u jednoj državi.*    ***Many-to-Many****– relacija po kojoj red u tabeli A može imati više povezanih redova u tabeli B, i obrnuto. Ovakva relacija se postiže kreiranjem zasebne tabele, koja se naziva povezujuća tabela, i koja sadrži strane ključeve iz obe povezane tabele A i B. Idealan primer ovakve relacije postoji između entiteta Author i Title. Naime, jedan autor može napisati veći broj naslova, a jedan naslov može biti potpisan od strane većeg broja autora.*    ***One-to-One****– relacija po kojoj red iz tabele A može imati ne više od jednog povezanog reda iz tabele B, i obrnuto. Ovakav tip relacija je generalno najređi, zato što se informacije modelovane na ovaj način gotovo uvek mogu objediniti unutar jedne tabele baze podataka. Ipak, ukoliko je potrebno tabelu sa velikim brojem kolona podeliti na dve manje, može se koristiti ovakav tip relacije. Često se ovakav tip relacija može susresti kod sistema koji su nadograđivani i gde zbog kompatibilnosti struktura postojećih tabela nije dirana, već je izvršeno samo nadograđivanje uvođenjem novih tabela.* |

**One-to-Many**

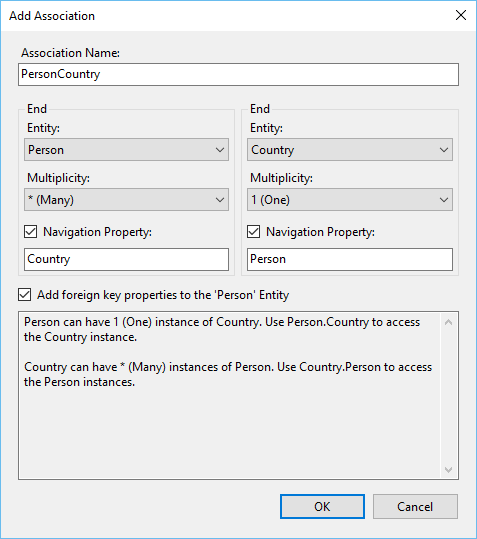
Kako bi se demonstrirala relacija One-to-Many, u postojećem primeru će biti dodat još jedan entitet sa nazivom *Country*, kao na slici 7.12.



*Slika 7.12 – Entitet Country*

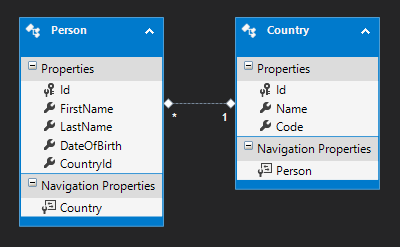
Tip relacije između entiteta *Person* i *Country* je jedan prema više.

Povezivanje entiteta se može izvršiti na sličan način na koji se vrši kreiranje istih. Potrebno je izvršiti desni klik na entitet i odabrati opciju *Add New -> Association*. Ovo kao efekat ima otvaranje prozora *Add Association* unutar koga je moguće izvršiti podešavanje povezivanja, što se vidi na slici 7.13.



*Slika 7.13 – Podešavanje povezivanja*

Nakon definisanja povezivanja izgled modela u okviru dizajnera je kao na slici 7.14.

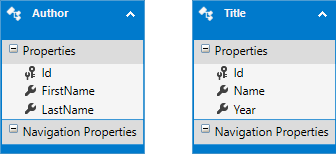


*Slika 7.14 – Povezani entiteti*

Odmah se može uočiti to da je entitet Person dobio novo polje *CountryId,* koje će sadržati informaciju o državi u kojoj osoba živi.

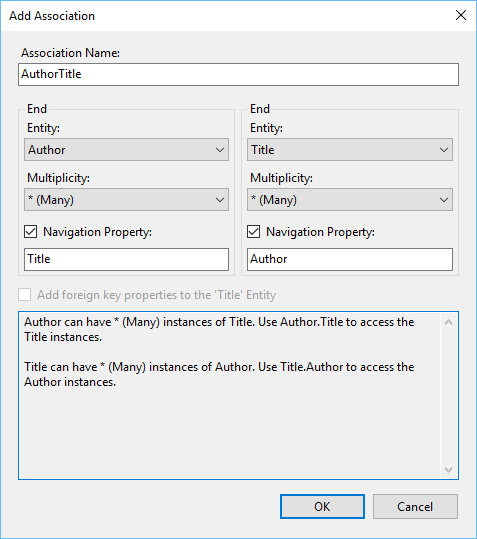
**Many-to-Many**

Relacija više prema više postoji između entiteta *Author* i *Book*. Struktura ovih entiteta je kao na slici 7.15.



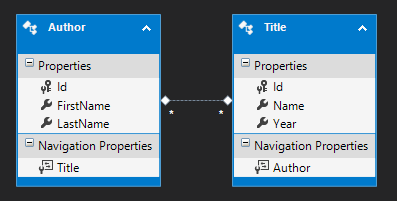
*Slika 7.15 – Npovezani Author i Title entiteti*

Definisanje relacije više prema više, prikazano je na slici 7.16.



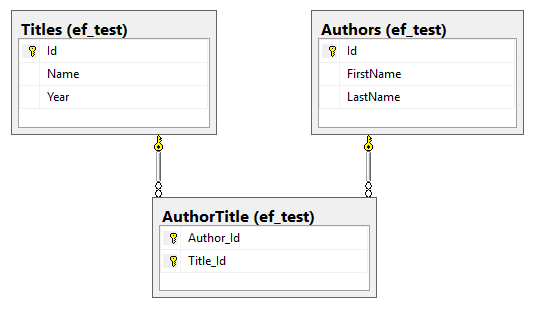
*Slika 7.16 – Podešavanje relacije više prema više*

Definisanjem relacije više prema više između dva entiteta, klase koje se na taj način generišu poseduju kolekcije objekata povezanog tipa. Tako klasa *Author* poseduje kolekciju objekata tipa *Title*, i obrnuto. Na taj način se relacija više prema više realizuje na objektom nivou. Povezani entiteti su prikazani slikom 7.17.



*Slika 7.17 – Povezani entiteti relacijom više prema više*

U bazi podataka situacija je drugačija. Relacija više prema više realizuje se kreiranjem dodatne tabele, koja se koristi za povezivanje. Dijagram tabela je prikazan slikom 7.18.

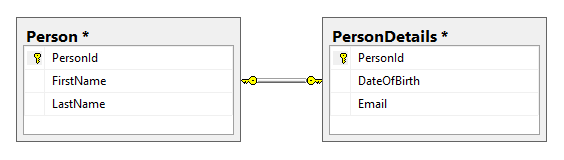


*Slika 7.18 – Dijagram tabela koje grade relaciju više prema više*

**Entity Splitting (spajanje više tabela u jedan entitet)**

Često se dešava da u bazi podataka postoji relacija One-to-One između dve tabele, koje modeluju podatke o jednom pojmu iz realnog sveta. U takvim situacijama potreba za postojanjem dva entiteta ne postoji, tako da se pribegava korišćenju tehnike *Entity Splitting*, koja podrazumeva deljanje jednog entiteta u više tabela, odnosno obrnuto.

Dijagram tabela baze podataka je kao na slici 7.19.

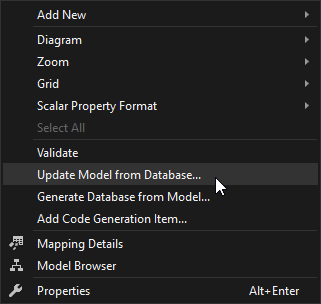


*Slika 7.19 – Tabele koje grade relaciju jedan prema jedan*

|  |
| --- |
| ***Napomena:****prilikom kreiranja prikazanih tabela u bazi podataka najbolje je za primarni ključ Person tabele postaviti*[*Auto Increment*](https://www.link-elearning.com/linkdl/opisPojma.php?id=142447)*svojstvo na true. Ipak, nikako se ne sme postaviti Auto Increment na true i za primarni ključ tabele PersonDetails, jer će se vrednost ovog ključa upisivati na osnovu vrednosti primarnog ključa Person tabele.* |

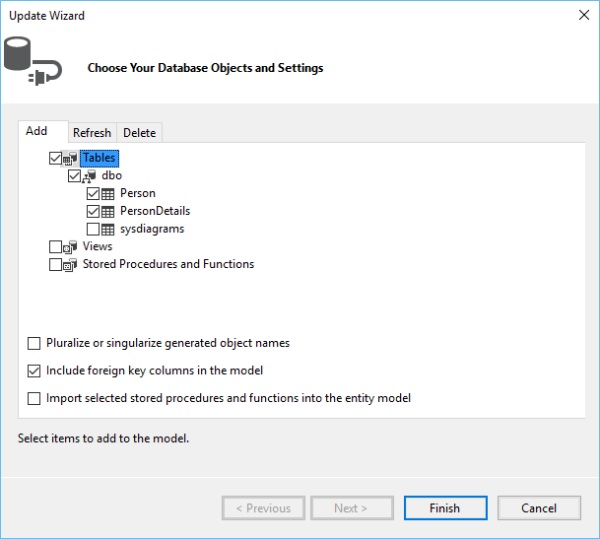
|  |
| --- |
| ***Napomena:****u prilogu lekcije se nalazi skripta pod nazivom one-to-one.sql za kreiranje prikazanih tabela baze podataka.* |

Prvo što je potrebno uraditi jeste kreiranje modela na osnovu prikazanih tabela baze podataka. To se može postići aktiviranjem opcije *Update Model from Database...,*kao na slici 7.20.



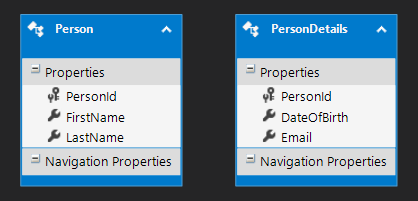
*Slika 7.20 – Opcija za ažuriranje modela na osnovu baze podataka*

U prozoru Update Wizard (slika 7.21) koji se otvorio potrebno je odabrati objekte baze podataka kojima je potrebno ažurirati model. To će u ovom primeru biti tabele *Person*i *PersonDetails*.



*Slika 7.21 – Odabir objekata baze podataka kojima će model biti dopunjen*

Sve ovo rezultuje kreiranjem dva entiteta, kao na slici 7.22.



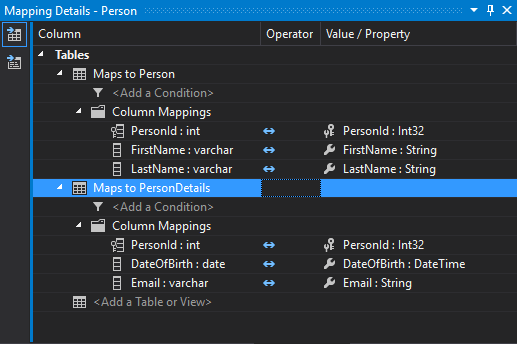
*Slika 7.22 – Dva entiteta koja inicijalno bivaju kreirana*

Prvi korak u spajanju entiteta podrazumeva fizičko prebacivanje svojstava iz entiteta *PersonDetails* u entitet *Person*. To se može postići selektovanjem svojstava *DateOfBirth* i *Email*, i njihovim premeštanjem u entitet Person. Za premeštanje se mogu koristiti prečice CTRL+X i CTRL+V.

Nakon prebacivanja ova dva svojstva u entitet Person, entitet PersonDetails je potrebno obrisati. Na pitanje koje sledi potrebno je odgovoriti sa No.

Sledeći korak podrazumeva ručno podešavanje mapiranja, koje se postiže korišćenjem prozora *Mapping Details*. Ovaj prozor se otvara odabirom opcije Mapping Details, koja se može pronaći unutar kontekstnog menija, koji se dobija desnim klikom na radnu površinu dizajnera.

Unutar prozora *Mapping Details* potrebno je izvršiti dodavanje još jedne tabele na koju se entitet mapira pritiskom na *<Add a Table or View>.*Za tabelu je potrebno odabrati *PersonInfo*. Nakon obavljanja svih opisanih koraka izgled prozora *Mapping Details*trebada bude kao na slici 7.23.



*Slika 7.23 – Podešavanje mapiranja entiteta Person*

Na slici 7.23 se može primetiti da je deo svojstava entiteta Person mapiran na kolone tabele *Person*, a deo na kolona tabele *PersonDetails*. To je upravo i efekat koji je bilo potrebno postići.

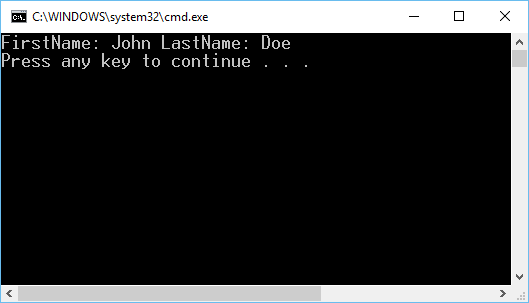
Na prikazan način, okruženje obavlja generisanje klase entiteta Person. Manipulacijom ovim tipom, potrebno je da podaci u bazi budu raščlanjeni na dve tabele. Da se to stvarno i obavlja, uverićemo se u nastavku lekcije.

Sledeći kod ilustruje predstavlja način na koji je moguće izvršiti upisivanje nove osobe u bazu podataka:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | using (var context = new EFTestContext())             {                 var person = new Person                 {                     FirstName = "John",                     LastName = "Doe",                     Email = "john@example.com",                     DateOfBirth = new DateTime(1988, 07, 13)                 };                 context.People.Add(person);                 context.SaveChanges();                 foreach (var per in context.People)                 {  Console.WriteLine("FirstName: " + person.FirstName + " LastName: " + person.LastName);                 }             } |

|  |
| --- |
| ***Napomena:****instanca tipa DataContext koja je u primeru imenovana promenljivom context zavisi od vrednosti svojstva Entity Container Name, postavljenim nešto ranije u okviru Properties prozora dizajnera.* |

Efekat prikazanog koda je kao na slici 7.24.



*Slika 7.24 – Efekat koda za unos nove osobe*

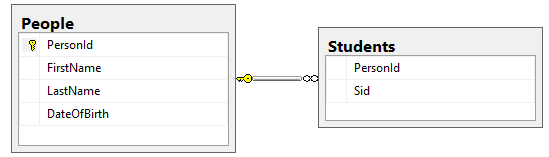
|  |
| --- |
| ***Napomena:****u navedenom primeru prikazano je na koji način se obavlja upisivanje entiteta u bazu podataka. To je inače materija koja će biti obrađena u narednom modulu, a ovde je prikazana samo demonstrativno.* |

**Modelovanje nasleđivanja**

Entiteti mogu naslediti osobine drugih entiteta, kao što klase mogu naslediti osobine drugih klasa. Entity Framework poznaje dva tipa nasleđivanja *Table per Type* i *Table per Hierarchy*.

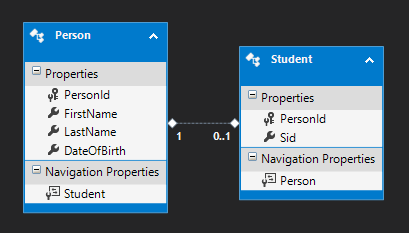
**Table per Type**

*Table per Type*(TPT) tip nasleđivanja podrazumeva da svaki tip koji učestvuje u relaciji nasleđivanja dobije posebnu tabelu u bazi podataka. Tako u bazi podataka može postojati situacija kao na slici 7.25.



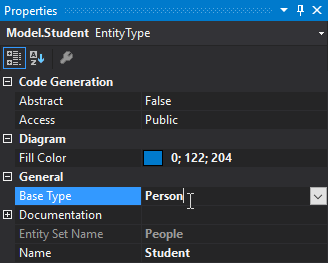
*Slika 7.25 – Nasleđivanje između dve tabele baze podataka*

Naredni primer prikazaće kako se može definisati nasleđivanje između entiteta Person i Student. Potrebno je kreirati model na osnovu baze podataka. To će kao rezultat imati kreiranje entiteta kao na slici 7.26.



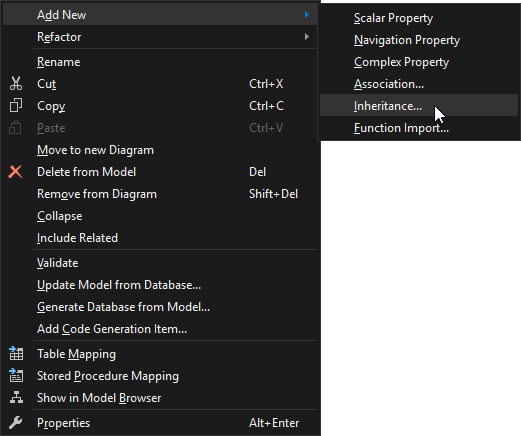
*Slika 7.26 – Nasleđivanje između dva entiteta*

Da bi se definisalo nasleđivanje među entitetima, potrebno je definisati roditeljski tip na tipovima potomcima. To znači da je potrebno za roditeljski tip entiteta Student postaviti tip Person. To se može postići u okviru Properties panela Student entiteta, kao na slici 7.27.



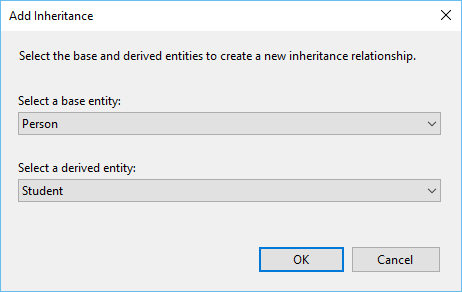
*Slika 7.27 – Definisanje roditeljskog tipa*

Takođe, nasleđivanje se može definisati i pokretanjem opcije *Add New -> Inheritance*... iz kontekstnog menija, koji se dobija desnim klikom na radnu površinu dizajnera, kao na slici 7.28.



*Slika 7.28 – Opcija za podešavanje nasleđivanja*

U prozoru koji se otvorio potrebno je odabrati osnovni i izvedeni entitet, kao na slici 7.29.

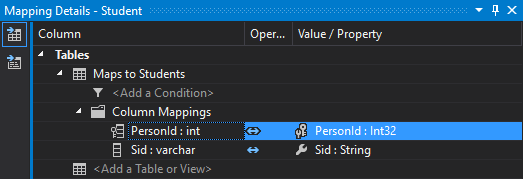


*Slika 7.29 – Prozor za podešavanje nasleđivanja*

Oba prikazana načina su ekvivalentna.

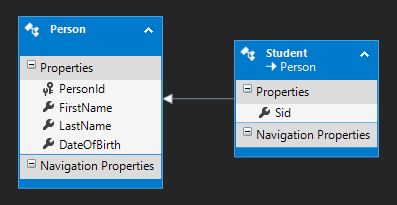
Nakon definisanja nasleđivanja potrebno je obrisati relaciju koju je okruženje automatski postavilo prilikom kreiranja modela na osnovu baze podataka.

Takođe je iz nasleđenih tipova potrebno obrisati Id svojstva, tako da će sledeći korak biti brisanje svojstva *PersonId* iz entiteta Student. Nakon brisanja svojstva *PersonId* potrebno je proveriti da li je kolona *PersonId* mapirana na odgovarajuće svojstvo, kao na slici 7.30.



*Slika 7.30 – Mapiranje kolona*

Nakon svih preduzetih operacija entiteti će izgledati kao na slici 7.31.



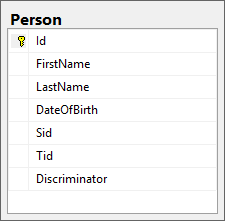
*Slika 7.31 – Nasleđivanje između dva entiteta*

Sada se kreirani tipovi modela mogu koristiti na sledeći način:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | using (var context = new EFTestContext())              {                  var student = new Student                  {                      FirstName = "John",                      LastName = "Doe",                      Sid = "027/44"                  };                  context.People.Add(student);                  context.SaveChanges();                  foreach (var per in context.People)                  {  Console.WriteLine("FirstName: " + student.FirstName + " LastName: " + student.LastName + " Sid: " + student.Sid);                  }              } |

**Table per Hierarchy**

Kod TPH tipa nasleđivanja koristi se jedna tabela u bazi podataka za smeštanje podataka svih entiteta. Entiteti se razlikuju korišćenjem diskriminatora. Tako tabela Person, koja bi se koristila za čuvanje podataka o entitetima Person, Student i Teacher, može da ima strukturu kao na slici 7.32.

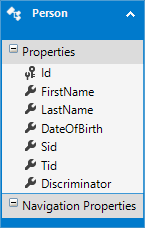


*Slika 7.32 – Tabela Person*

Kolone *Id*, *FirstName*, *LastName* i *DateOfBirth* su obavezne za sve osobe, bilo da su one studenti ili predavači. Kolona *Sid* (Student Identification Number) je karakteristična samo za studente, a kolona *Tid* (Teacher Identification Number) je karakteristična samo za predavače. Na kraju, kolona Discriminator se koristi kako bi definisala tip zapisa. Ova kolona može imati dve vrednosti, i to *student* i *teacher*, na osnovu kojih se određuje tip zapisa.

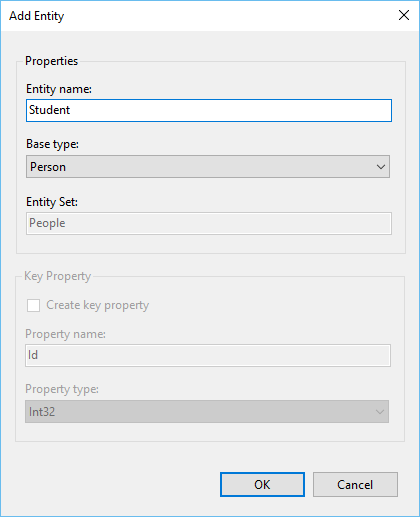
Kako bi se ovakva tabela baze podataka pretočila u model sa entitetima, potrebno je uraditi sledeće.

Kreiranjem modela na osnovu baze podataka inicijalno se dobija jedan entitet, kao na slici 7.33.

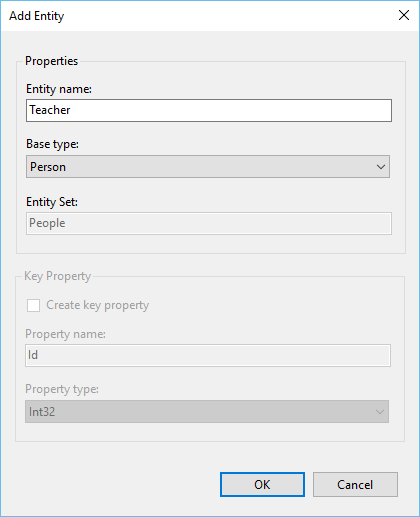


*Slika 7.33 – Inicijalni entitet Person*

1. Prvi korak u kreiranju entiteta sa odgovarajućim odnosom nasleđivanja jeste kreiranje entiteta Student i Teacher, i postavljanje njihovog roditeljskog tipa. Slike 7.34 i 7.35 dočaravaju dijaloge za dodavanje entiteta i definisanje roditeljskog tipa.

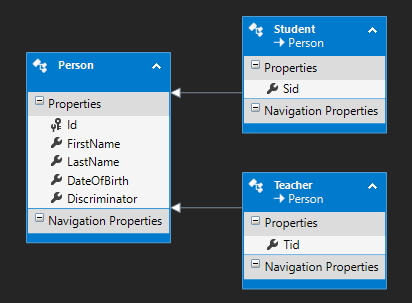


*Slika 7.34 – Dodavanje entiteta Student*



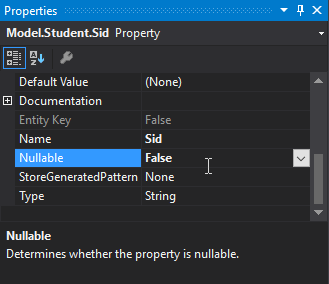
*Slika 7.35 – Dodavanje entiteta Teacher*

2. Drugi korak podrazumeva premeštanje svojstava koja su karakteristična za izvedene tipove iz roditeljskog tipa u konkretne tipove, kao na slici 7.36.



*Slika 7.36 – Prebacivanje svojstava karakterističnih za Student i Teacher entitete*

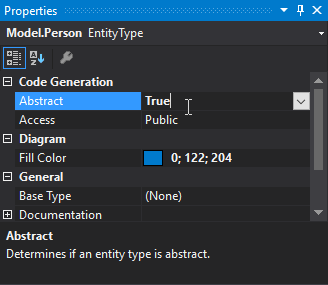
3. Treći korak podrazumeva definisanje toga da *Sid* i *Tid* svojstva ne mogu da prihvate null vrednosti. To se postiže seletovanjem svojstva i odabirom opcije Properties. U okviru Properties panela dovoljno je postaviti vrednost svojstva Nullable na false, kao na slici 7.37.



*Slika 7.37 – Definisanje vrednosti za svojstvo Nullable*

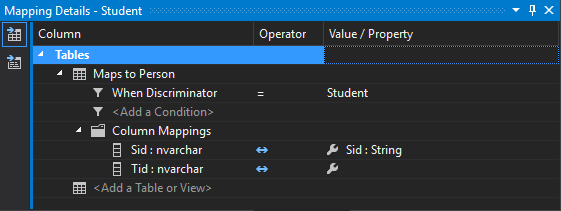
4. Iz entiteta Person potrebno je ukloniti svojstvo Discriminator. Nešto kasnije biće objašnjen razlog uklanjanja ovog svojstva.

5. Nakon definisanja svih entiteta i njihovih svojstava, kao i međusobnih relacija, potrebno je entitet Person markirati kao apstraktni, što vidimo na slici 7.38.

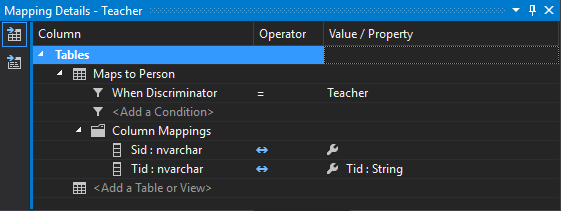


*Slika 7.38 – Definisanje apstraktnog entiteta*

6. Nakon podešavanja entiteta potrebno je preći na mapiranje entiteta. Po prvi put u ovom kursu biće upotrebljeno uslovno mapiranje. Uslovno mapiranje će biti obavljeno na osnovu Discriminator kolone, kao na slikama 7.39 i 7.40.



*Slika 7.39 – Uslovno mapiranje entiteta Student*

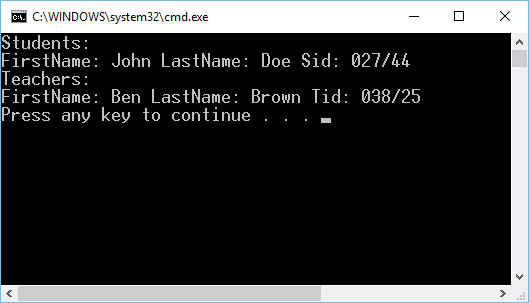


*Slika 7.40 – Uslovno mapiranje entiteta Teacher*

Sledeći kod ilustruje upotrebu tipova koji su kreirani na ovaj način:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | using (var context = new EFTestContext())              {                  var student = new Student                  {                      FirstName = "John",                      LastName = "Doe",                      DateOfBirth = "1988/07/13",                      Sid = "027/44"                  };                  context.People.Add(student);                  var teacher = new Teacher                  {                      FirstName = "Ben",                      LastName = "Brown",                      DateOfBirth = "1988/05/05",                      Tid = "038/25"                  };                  context.People.Add(teacher);                  context.SaveChanges();                  Console.WriteLine("Students:");                  foreach (var stud in context.People.OfType<Student>())                  {                      Console.WriteLine("FirstName: " + stud.FirstName + " LastName: " + stud.LastName + " Sid: " + stud.Sid);                  }                  Console.WriteLine("Teachers:");                  foreach (var teach in context.People.OfType<Teacher>())                  {                      Console.WriteLine("FirstName: " + teach.FirstName + " LastName: " + teach.LastName + " Tid: " + teach.Tid);                  }              } |

Nakon izvršavanja prikazanog koda rezultat obrade će biti kao na slici 7.41.



*Slika 7.41 – Zapisi pročitani iz baze podataka*

|  |
| --- |
| ***Napomena:****razlog zbog kojeg je uklonjeno svojstvo Discriminator iz entiteta Person je taj što nije moguće izvršiti mapiranje kolone tabele više od jednom. S obzirom na to da se ova kolona koristi za kondicionalno mapiranje, nju nije moguće koristiti i za mapiranje svojstava.* |