**Uvod u Code First**

**00:13:34**

Jedinica: 14 od 19

**Rezime**

* Korišćenjem Code First pristupa, akcenat se stavlja na dizajniranje domenskog modela, kreiranjem klasa koje će predstavljati entitete.
* Code Frist API samostalno obavlja kreiranje baze podataka.
* DbContext automatski kreira bazu podataka, i to prvo pokušava da uradi korišćenjem instance  SQL Express servera, koja je instalirana zajedno sa Visual Studio razvojnim okruženjem. Ako ova instanca nije aktivna, DbContext će koristiti LocalDB.
* Ukoliko se drugačije ne navede, naziv baze podataka biće kreiran na osnovu punog naziva kontekstne klase, u koji ulazi naziv same klase, ali i nazivi prostora imena u kojima se takva klasa nalazi.
* EF Code First API kreira tabele baze podataka na osnovu svojstava tipa DbSet koja se nalaze u kontekstnoj klasi.
* Podrazumevana konvencija je takva da se za primarni ključ uzima svojstvo sa nazivom *Id* ili sa nazivom *<naziv klase>Id*, gde se naziv klase odnosi na naziv klase entiteta, pri čemu se velika i mala slova tretiraju jednako.

U dosadašnjem toku kursa, u svim primerima, upotreba EF modela se svodila na korišćenje Model First ili Database First pristupa. Jednostavno, domenski model je generisan na osnovu baze podataka ili na osnovu modela načinjenog korišćenjem EDM dizajnera. U narednim lekcijama pažnja će biti posvećena preostalom od tri pristupa – Code First pristupu.

**Šta je Code First?**

Korišćenjem Code First pristupa, akcenat je na dizajniranju domenskog modela, kreiranjem klasa koje će predstavljati entitete. Tako se kreiranje baze podataka prepušta Code First API-u.

Slika 14.1 dočarava postupak u kome se Code First pristupom kod koristi za kreiranje konceptualnog modela.



*Slika 14.1 – Code First pristup*

Iz ugla programera, korišćenje Code First pristupa podrazumeva pisanje C# ili VB.NET klasa, kao i kontekstne klase. Prilikom prvog pokretanja aplikacije, Code First API, sastavni deo EF-a, vrši kreiranje baze podataka na osnovu kreiranih klasa, kao i odgovarajuća mapiranja. Tako bi neki ustaljeni flow prilikom korišćenja Code First modela podrazumevao sledeće korake:

* + pisanje domenskih klasa i klase konteksta,
  + konfigurisanje klasa i mapiranja,
  + generisanje tabela baze.

U nastavku lekcije biće prikazana jednostavna demonstracija korišćenja Code First pristupa.

|  |
| --- |
| ***Napomena:****za automatsko dodavanje svih biblioteka potrebnih za rad EF-a, ali i odgovarajućeg konfiguracionog fajla, najbolje je odmah na početku to obaviti korišćenjem NuGeta i to sledećom komandom:*    *Install-Package EntityFramework.* |

**Kreiranje klasa modela**

Prvi korak u sprovođenju Code First pristupa podrazumeva kreiranje klasa modela. U primeru u ovoj lekciji biće kreirane dve klase: *Person* i *Country*.

Klasa Person:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | class Person  {      public int PersonId { get; set; }      public string FirstName { get; set; }      public string LastName { get; set; }      public DateTime DateOfBirth { get; set; }      public Country Country { get; set; }  } |

Klasa Country:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | class Country  {      public int CountryId { get; set; }      public string Name { get; set; }      public string Capital { get; set; }      public int Population { get; set; }      public int MyProperty { get; set; }      public ICollection<Person> People { get; set; }  } |

**Kreiranje kontekstne klase**

Nakon kreiranja klasa domenskog modela, Code First pristup zahteva kreiranje i kontekstne klase, koja je potrebno da nasledi klasu DbContext. Klasa će imati sledeći izgled:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | class PersonContext : DbContext      {          public PersonContext() : base()          {            }          public DbSet<Person> People { get; set; }          public DbSet<Country> Countries { get; set; }      } |

|  |
| --- |
| ***Napomena:****za korišćenje klase DbContext, potrebno je u tekući dokument uključiti prostor imena System.Data.Entity.* |

Kontekstna klasa je potrebno da sadrži kolekcije entiteta kojima će rukovati. Tako klasa PersonContext sadrži dve kolekcije tipa DbSet. Jedna se koristi za čuvanje osoba, a druga za čuvanje država.

**Korišćenje kreiranih domenskih klasa**

Kreiranjem klasa entiteta, kao i klase koja će služiti kao kontekst, završeno je kreiranje domenskog modela. Sledeći blok koda ilustruje na koji način se kreirane klase upotrebljavaju.

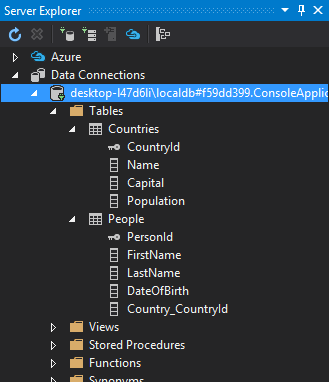
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | using (var ctx = new PersonContext())  {      Person person = new Person();      person.FirstName = "John";      person.DateOfBirth = new DateTime(1941, 04, 05);      ctx.People.Add(person);      ctx.SaveChanges();  } |

Unutar jednog using bloka instancirana je kontekstna klasa. Kreiran je jedan objekat tipa Person odnosno, kreiran je jedan entitet. Entitet je dodat kontekstu korišćenjem kolekcija tipa DbSet, koje su nešto ranije kreirane unutar kontekstne klase. Na kraju, pozivanjem metode SaveChanges, ove promene su perzistirane. Pokretanjem prikazanog koda, može se zaključiti da on prolazi bez emitovanja bilo kakve greške ili poruke.

**Šta je sa podacima?**

Sve što je do sada prikazano u ovoj lekciji sasvim je dovoljno kako bi se obavilo generisanje baze podataka. Konektovanjem na skladište može se zaključiti da je EF Code First API uspešno izvršio kreiranje baze podataka.

Takođe, konektovanjem na skladište, gde je okruženje izvršilo generisanje klasa, može se steći uvid u strukturu tabela, kao na slici 14.2.



*Slika 14.2 – Code First pristup*

Sada se postavlja pitanje kako je okruženje odabralo server baze podataka i na koji način je formiralo naziv kreirane baze.

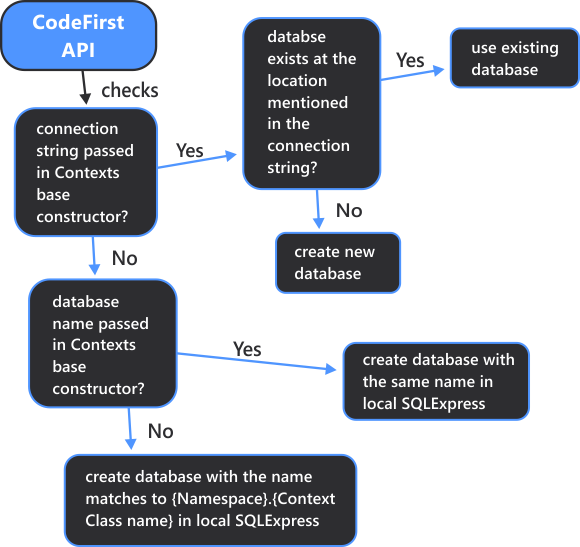
**Inicijalizacija baze podataka**

Po pravilu, DbContext automatski kreira bazu podataka, i to prvo pokušava da uradi korišćenjem instance  [SQL Express servera](https://www.link-elearning.com/linkdl/opisPojma.php?id=142504), koja je instalirana zajedno sa Visual Studio razvojnim okruženjem. Ako ova instanca nije aktivna, DbContext će koristiti [LocalDB](https://www.link-elearning.com/linkdl/opisPojma.php?id=142454).

|  |
| --- |
| ***Napomena:****ukoliko je potrebno povezati se sa LocalDB skladištem, prilikom kreiranja konekcije za naziv servera dovoljno je uneti: (localdb)\MSSQLLocalDB.* |

Naziv baze podataka biće kreiran na osnovu punog naziva kontekstne klase u koji ulazi naziv same klase, ali i nazivi prostora imena u kojima se takva klasa nalazi.

Sve do sada rečeno odnosi se samo na situacije kada se sistem uposli bez bilo kakvih dodatnih podešavanja. Ipak, EF omogućava eksplicitno definisanje naziva baze, kao i servera na kome će takva baza biti kreirana. Na koji način EF Code First API odlučuje o tome šta će uraditi, prikazano je na slici 14.3.



*Slika 14.3 – Postupak po kojem EF Code First API odlučuje gde će kreirati bazu podataka*

Analizom slike 14.3 može se zaključiti da je konstruktoru kontekste klase moguće prosleđivati izvesne parametre kojima bi se uticalo na naziv baze i mesto na kome će ona biti generisana. Tako se konstruktor kontekstne klase može koristiti:

* + bez parametara,
  + sa parametrom koji predstavlja naziv baze,
  + sa parametrom koji predstavlja konekcioni string.

***Konstruktor bez parametara***

Ukoliko se ne navede nijedan parametar konstruktoru osnovne klase, okruženje kreira bazu podataka na lokalnoj instanci SQLEXPRESS servera ili u okviru LocalDb-a, kao što je već rečeno. Naziv baze se u takvom slučaju formira po obrascu:

{Namespace}.{Context class name}

***Naziv baze kao parametar konstruktora***

Moguće je kao parametar konstruktora kontekstne klase definisati naziv baze podataka. U takvoj situaciji, EF Code First API neće koristiti već pomenuti obrazac za formiranje imena baze, već će iskoristiti zadati naziv. Primer definisanja naziva unutar konstruktora kontekstne klase je sledeći:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | class PersonContext : DbContext  {      public PersonContext() : base("EfTestDb")      {      }  } |

***Konekcioni string kao parametar konstruktora***

Prilikom korišćenja Code First pristupa, moguće je definisati i konekcioni string unutar app.cofig ili web.config fajla, a zatim naziv tako definisanog konekcionog stringa koristiti kao parametar konstruktora kontekstne klase. Takav parametar se definiše kao string sa prefiksom *name=*. Sledeći primer to i ilustruje:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | class PersonContext : DbContext  {      public PersonContext() : base("name = TestConnString")      {      }      ...  } |

U osnovnom (base) konstruktoru izvedene DbContext klase, kao parametar naveden je konstruktor sa nazivom TestConnString. Stoga je unutar konfiguracionog fajla potrebno imati sledeće:

|  |  |
| --- | --- |
| 1  2  3 | <connectionStrings>    <add name="TestConnString" connectionString="Data Source=localhost;Initial Catalog=TestEF;Integrated Security=SSPI" providerName="System.Data.SqlClient"/>  </connectionStrings> |

Na ovaj način programer ima potpunu slobodu po pitanju lokacije na kojoj će biti generisana baza podataka. Tako je, korišćenjem navedenog pristupa, moguće kreirati bazu podataka i u okviru instance pune verzije SQL Servera.

**Podrazumevano, ukoliko se upotrebi konstruktor bez parametara, EF prvo pokušava da kreira bazu:**

 na lokalnoj instanci SQLExpress servera

 na lokalnoj instanci punog SQL servera

 unutar sistema LocalDB

 unutar tekstualnog fajla u root direktorijumu projekta

**Konvencije koje se koriste upotrebom Code First pristupa**

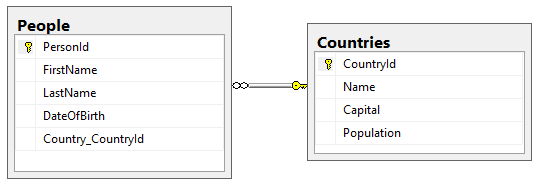
Generalno govoreći, konvencija jeste skup pravila. Kada se govori o Code First pristupu, pod pojmom konvencija smatraju se pravila koja se koriste za automatsko konfigurisanje konceptualnog modela na osnovu domenskih klasa. Takve konvencije definisane su u prostoru imena System.Data.Entity.ModelConfiguration.Conventions.

**Tabele**

Prva konvencija o kojoj će biti reči jeste ona po kojoj se kreiraju tabele baze podataka. EF Code First API kreira tabele baze podataka na osnovu svojstava tipa DbSet, koja se nalaze u kontekstnoj klasi. Tako se u kontekstnoj klasi u primeru ove lekcije nalaze sledeće dva svojstva tipa DbSet:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | class PersonContext : DbContext  {      public PersonContext() : base("name = TestConnString")      {      }      public DbSet<Person> People { get; set; }      public DbSet<Country> Countries { get; set; }  } |

Na osnovu ovih svojstava kreirane su tabele, kao na slici 14.4.



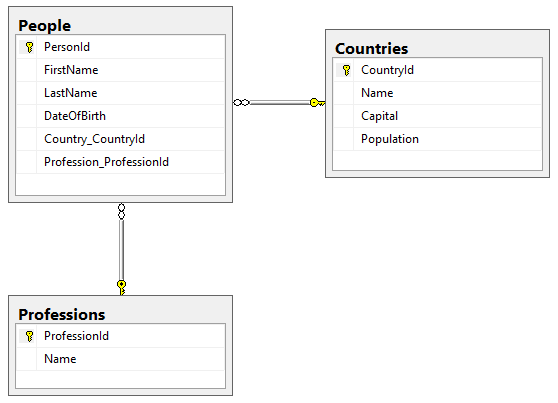
*Slika 14.4 – Tabele kreirane na osnovu koda korišćenjem EF Code First API-a*

EF Code First API izvršiće generisanje tabela i na osnovu tipova koji su referencirani, pa čak i ako kolekcije takvih tipova nisu definisane unutar kontekstne klase. Sledeći primer će to ilustrovati.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | class Person  {      public int PersonId { get; set; }      public string FirstName { get; set; }      public string LastName { get; set; }      public DateTime DateOfBirth { get; set; }      public Country Country { get; set; }      public Profession Profession { get; set; }  } |

Klasi Person dodato je svojstvo tipa Profession.

Nakon prvog pokretanja aplikacije situacija u bazi će biti kao na slici 14.5.



*Slika 14.5 – Tabela Professions koja je kreirana na osnovu referenciranog tipa*

**Primarni ključevi**

EF Code First API automatski kreira i primarne ključeve za svaku tabelu. Podrazumevana konvencija je takva da se za primarni ključ uzima svojstvo sa nazivom *Id* ili sa nazivom *<naziv klase>Id*, gde se naziv klase odnosi na naziv klase entiteta, pri čemu se velika i mala slova tretiraju jednako. Tip svojstva koje će da predstavlja primarni ključ može biti bilo šta, a ukoliko je takav tip numerički ili GUID, kolona primarnog ključa biće konfigurisana kao kolona identiteta, što znači da će imati [Auto-Increment](https://www.link-elearning.com/linkdl/opisPojma.php?id=142447) svojstvo.

Ukoliko se naziv kolone primarnog ključa definiše na drugačiji način, okruženje će izbaciti izuzetak tipa ModelValidationException. Ukoliko baš postoji potreba za drugačijim imenovanjem svojstva primarnog ključa, moguće je obaviti konfigurisanje klasa modela korišćenjem anotacija ili FluentAPI-a, o čemu će više reči biti u narednoj lekciji.

**Relacije**

EF Code First API detektuje relacije između entiteta na osnovu takozvanih navigacionih svojstava. Takva svojstva su jednostavne reference na povezane entitete ili kolekcije entiteta. Na primer, u klasi Person su definisana dva takva svojstva, i ona su sledeća:

|  |  |
| --- | --- |
| 1  2 | public Country Country { get; set; }  public Profession Profession { get; set; } |

U klasi Country postoji jedno takvo svojstvo i ono je sledeće:

|  |  |
| --- | --- |
| 1 | public ICollection<Person> People { get; set; } |

Ovakve dve tabele u konceptualnom modelu povezane su jednim stranim ključem sa nazivom: *Country\_CountryId*. Tako je podrazumevana konvencija za imenovanje stranih ključeva sledeća:

<navigation property Name>\_<primary key property name of navigation property type>