**CRUD operacije**

**00:12:16**

Jedinica: 12 od 19

**Rezime**

* Akronim CRUD se odnosi na pojmove CREATE, READ, UPDATE i DELETE.
* Klasa DBSet predstavlja kolekciju entiteta, a koristi se za vršenje operacija kreiranja, čitanja, ažuriranja i brisanja podataka.
* Za dodavanje entiteta koristi se metoda Add, DbSet klase.
* Klasa **DbChangeTracker** omogućava čitanje svih informacija o entitetima koji se prate unutar konteksta.
* Ažuriranje entiteta i njegovih osobina se može postići modifikovanjem svojstava objekta entiteta i pozivanjem metode SaveChanges.
* Brisanje entiteta podrazumeva dobavljanje reference na entitet koji je potrebno obrisati i pozivanje metode Remove nad kolekcijom entiteta, tipa DbSet.

U prethodnim lekcijama uglavnom je bilo prikazano kako se podaci čitaju, odnosno pregledaju iz baze podataka. To je ipak samo jedna od uobičajenih operacija nad podacima, koje su objedinjene akronimom CRUD.

Akronim CRUD se odnosi na pojmove **C**REATE, **R**EAD, **U**PDATE i **D**ELETE. Tako će lekcija pred Vama biti posvećena tehnikama za upravljanje podacima, koje podrazumevaju kreiranje, modifikovanje i brisanje.

**DBSet klasa**

Klasa DBSet predstavlja kolekciju entiteta, a koristi se za vršenje operacija kreiranja, čitanja, ažuriranja i brisanja podataka. Generička verzija ove klase DbSet<TEntity> može biti korišćena kada je tip entiteta nepoznat prilikom kompajliranja.

Referenca na objekte ove klase se može dobiti korišćenjem objekta klase [DBContext](https://www.link-elearning.com/linkdl/opisPojma.php?id=142493), kao u sledećem primeru:

|  |  |
| --- | --- |
| 1  2  3  4 | using (var ctx = new NorthwindEntities())  {      DbSet customers = ctx.Customers;  } |

Ukoliko se pogleda definicija klase koja nasleđuje klasu DbContext, može se primetiti mnoštvo svojstava tipa DbSet, što se vidi na slici 12.1.



*Slika 12.1 – Svojstva tipa DbSet koja se nalaze unutar kontekstne klase*

Neke od najznačajnijih metoda klase DbSet prikazane su tabelom 12.1.

|  |  |
| --- | --- |
| **Metoda** | **Opis** |
| **Add** | Dodaje entitet sa stanjem Added u kontekst. Kada se promene sačuvaju, ovakvi entiteti sa stanjem Added ubacuju se u bazu podataka. Nakon čuvanja stanje objekta se menja u Unchanged. |
| **AsNoTracking** | Entiteti vraćeni od strane ove metode neće biti praćeni od strane objekta DBContext. Ovako nešto može da ima značajan pozitivan uticaj na performanse entiteta koji su predviđeni samo za čitanje. |
| **Attach** | Priključuje entitet sa stanjem Unchanged kontekstu. |
| **Create** | Kreira novu instancu entiteta onog tipa koji odgovara setu podataka, ali se takva instanca ne dodaje, niti priključuje setu. |
| **Find(int)** | Koristi vrednost primarnog ključa kao ulazni parametar za pronalaženje entiteta koji se je u kontekstu. Ukoliko se traženi entitet ne pronađe u kontekstu, pretraga se prebacuje na bazu podataka. |
| **Include** | Omogućava uključivanje LINQ to Entities upita. |
| **Remove** | Markira prosleđeni entitet kao obrisan. Kada se promene sačuvaju, entitet se stvarno i briše iz baze. Neophodno je da entitet postoji unutar konteksta u bilo kojem drugom stanju, pre nego što se ova metoda pozove. |
| **SqlQuery** | Omogućava kreiranje sirove SQL naredbe. Podrazumevano, entiteti koji se dobiju na ovaj način ulaze u kontekst i bivaju praćeni. |

*Tabela 12.1 – Najznačajnije metode klase DbSet*

**Dodavanje entiteta**

Kao što je to u dosadašnjem toku lekcije rečeno, za dodavanje entiteta koristi se metoda Add, DbSet klase.

|  |
| --- |
| ***Napomena:****u jednoj od prethodnih lekcija objašnjena je razlika između klasa ObjectContext i DbContext. Na isti način EF poznaje i dva različita skupa tipova koja se koriste za predstavljanje kolekcija entiteta i njihovih svojstava. Tako je ekvivalent tipa DbSet, tip ObjectSet.* |

Potpis metode Add je sledeći:

|  |  |
| --- | --- |
| 1 | public Object Add( Object entity) |

Sledi primer dodavanja objekta u kontekstni objekat. Prvo je potrebno kreirati novi objekat koji se dodaje. U primeru će to biti objekat tipa Employee.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | using (var ctx = new NorthwindEntities())  {      Employee emp = new Employee      {          Address = "Hight Street",          BirthDate = DateTime.Today.AddYears(-30),          City = "New York",          Country = "New York",          FirstName = "Mark",          LastName = "Popovich",          Title = "Mr",          TitleOfCourtesy = "Sir",          HireDate = DateTime.Today.AddYears(-3)      };      ctx.Employees.Add(emp);      ctx.SaveChanges();  } |

Na samom kraju, kao što se to može videti, pomoću metode SaveChanges obavlja se čuvanje svih promena u bazi podataka. Vrlo bitno je razumeti da se promene napravljene na objektnom modelu neće preslikati na bazu podataka automatski, bez prethodnog pozivanja ove metode.

|  |
| --- |
| ***Napomena:****SaveChanges metoda definisana je u prostoru imena System.Data.Objects.* |

Za proveru uspešnosti unosa novog zapisa, može se napisati nešto ovako:

|  |  |
| --- | --- |
| 1 | Console.WriteLine("Employee ID : {0}", emp.EmployeeID); |

Ovakva linija vratiće Id vrednost koju je zapis dobio u bazi podataka. Zapis koji je na ovaj način kreiran u bazi imaće prvi naredni slobodan Id. Razlog ovome je *Auto Increment* funkcionalnost, koja se primenjuje na primarnim ključevima Northwind tabela.

|  |
| --- |
| ***Napomena:****Entity Framework generiše privremeni ključ za svaki novi objekat,  u trenutku kreiranja. Nakon poziva SaveChanges metode stavka se smešta u bazu, i dodeljuje joj se Id. Ukoliko baza nije konfigurisana da generiše vrednosti ključa, potrebno je vrednost ključa samostalno dodeliti. Ukoliko više objekata ima istu vrednost ključa, nastaće InvalidOperationException izuzetak, prilikom poziva metode SaveChanges.* |

**Koji izuzetak može da se javi prilikom čuvanja entiteta u bazi?**

 InvalidOperationException

 InvalidException

 OperationException

 InvalidOperation

**Praćenje promena**

Entity Framework omogućava automatsko praćenje promena učitanih entiteta tokom životnog veka konteksta. Klasa **DbChangeTracker** omogućava čitanje svih informacija o entitetima koji se prate unutar konteksta.

|  |
| --- |
| ***Napomena:****vrlo je bitna informacija da svaki entitet mora imati primarni ključ, kako bi mogao da bude praćen. EF neće izvršiti dodavanje entiteta unutar konceptualnog modela koji ne poseduje primarni ključ.* |

Sledeća metoda enkapsulira kod za ispis svih objekata koji se prate, kao i njihovog trenutnog stanja:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | public static void DisplayTrackedContextEntities(DbContext ctx)  {      DbChangeTracker changeTracker = ctx.ChangeTracker;      var entries = changeTracker.Entries();      Console.WriteLine("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");      foreach (var entry in entries)      {          Console.WriteLine("Entity Name: {0}", entry.Entity.GetType().Name);          Console.WriteLine("Status: {0}", entry.State);      }      Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");  } |

Prikazana DisplayTrackedContextEntities metoda koristiće se u narednim primeru, da bi dočarala promene stanja kroz koje praćeni objekti prolaze.

|  |  |
| --- | --- |
| 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  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57 | class Program      {          static void Main(string[] args)          {              using (var ctx = new NorthwindEntities())              {                    //adding entity to context                  Employee employeeEntity = ctx.Employees.Find(15);                  DisplayTrackedContextEntities(ctx);                  //adding one more entity to context                  Employee employeeEntity2 = ctx.Employees.Find(14);                  DisplayTrackedContextEntities(ctx);                  //modifying entity                  employeeEntity.FirstName = "ChangedName";                  DisplayTrackedContextEntities(ctx);                  //detach entity                  ((IObjectContextAdapter)ctx).ObjectContext.Detach(employeeEntity);                  DisplayTrackedContextEntities(ctx);                  //creating entity                  Employee emp = new Employee                  {                      Address = "Hight Street",                      BirthDate = DateTime.Today.AddYears(-30),                      City = "New York",                      Country = "New York",                      FirstName = "Mark",                      LastName = "Popovich",                      Title = "Mr",                      TitleOfCourtesy = "Sir",                      HireDate = DateTime.Today.AddYears(-3)                  };                  ctx.Employees.Add(emp);                  DisplayTrackedContextEntities(ctx);                      //saving entity                  ctx.SaveChanges();                  DisplayTrackedContextEntities(ctx);                  //removing entity                  ctx.Employees.Remove(emp);                  DisplayTrackedContextEntities(ctx);              }          }          public static void DisplayTrackedContextEntities(DbContext ctx)          {              DbChangeTracker changeTracker = ctx.ChangeTracker;              var entries = changeTracker.Entries();              Console.WriteLine("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");              foreach (var entry in entries)              {                  Console.WriteLine("Entity Name: {0}", entry.Entity.GetType().Name);                  Console.WriteLine("Status: {0}", entry.State);              }              Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");          }      } |

Prikazani primer proizvodi sledeći izlaz:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Entity Name: Employee  
5F8329  
Status: Unchanged  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Entity Name: Employee  
5F8329  
Status: Unchanged  
Entity Name: Employee  
5F8329  
Status: Unchanged  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Entity Name: Employee  
5F8329  
Status: Modified  
Entity Name: Employee  
5F8329  
Status: Unchanged  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Entity Name: Employee  
5F8329  
Status: Unchanged  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Entity Name: Employee  
Status: Added  
Entity Name: Employee  
5F8329  
Status: Unchanged  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Entity Name: Employee  
Status: Unchanged  
Entity Name: Employee  
5F8329  
Status: Unchanged  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
Entity Name: Employee  
Status: Deleted  
Entity Name: Employee  
5F8329  
Status: Unchanged  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Ažuriranje entiteta (Update)**

Ažuriranje entiteta i njegovih osobina se može postići vrlo jednostavno, modifikovanjem svojstava objekta entiteta i pozivanjem metode SaveChanges.

Sledeći primer ilustruje način na koji se obavlja ažuriranje zapisa o jednom zaposlenom.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | using (var ctx = new NorthwindEntities())  {      var emp = (from e in ctx.Employees where e.EmployeeID == 15 select e).First();      emp.LastName = "NewLastName";      emp.Address = "NewAddress";      ctx.SaveChanges();  } |

**Brisanje entiteta (Delete)**

Brisanje entiteta podrazumeva dobavljanje reference na entitet koji je potrebno obrisati i pozivanje metode Remove nad kolekcijom entiteta tipa DbSet.

Sledeći primer ilustruje brisanje entiteta.

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | using (var ctx = new NorthwindEntities())            {  var emp = (from e in ctx.Employees where e.EmployeeID == 15 select e).First();                ctx.Employees.Remove(emp);                ctx.SaveChanges();            } |