**Uvod u rad sa podacima**

**00:20:47**

Jedinica: 1 od 19

**Rezime**

* Perzistencija je pojam koji se odnosi na mogućnost čuvanja objekata i nakon prestanka izvršavanja aplikacije.
* Perzistencija se najčešće obavlja čuvanjem podataka u tekstualnim fajlovima ili bazama podataka.
* Baza podataka je kolekcija strukturiranih informacija koja upravlja velikom količinom podataka i na organizovan i strukturiran način omogućava korisniku rukovanje podacima.
* Sistem za upravljanje relacionom bazom podataka naziva se Relational DataBase Management System ili skraćeno RDBMS.
* ADO.NET je set klasa koje izlažu servise za pristup podacima .NET programerima, obezbeđujući na taj način bogat set komponenti za kreiranje distribuiranih i sadržajno bogatih aplikacija.
* Objektno-relaciono mapiranje je pojam koji se odnosi na programske tehnike koje obavljaju konvertovanje podataka između nekompatibilnih sistema za predstavljanje podataka.
* ADO.NET Entity Framework je ultimativna Microsoft tehnologija za razvoj aplikacija koje rukuju podacima.
* LINQ omogućava standardni, objedinjeni pristup za pretragu i modifikovanje podataka koji može da podrži praktično bilo koji tip podataka.

Rad sa podacima predstavlja osnovu gotovo svake aplikacije, bez obzira na njen tip i veličinu. U kursevima koji su se bavili razvojem desktop i web aplikacija korišćenjem Microsoft tehnologija, prikazano je kako se podacima rukuje korišćenjem različitih tipova aplikacija. Sada je vreme da se ovom veoma značajnom pojmu posveti znatno više pažnje. Stoga će u lekcijama koje slede biti obrađene tehnike za upravljanje podacima kroz aplikacije koje se razvijaju korišćenjem [Microsoft](https://www.link-elearning.com/linkdl/opisPojma.php?id=142473)tehnologija.

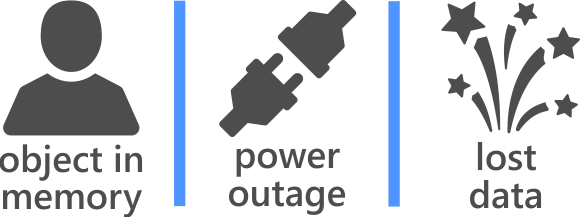
Kada se kaže da je aplikacija sposobna da čuva podatke, prevashodno se misli na čuvanje koje nadživljava životni vek takve aplikacije. Takav postupak drugačije sa naziva i perzistencija.

**Šta je perzistencija?**

Pojam perzistencije se odnosi na mogućnost čuvanja objekta i nakon prestanka izvršavanja aplikacije.

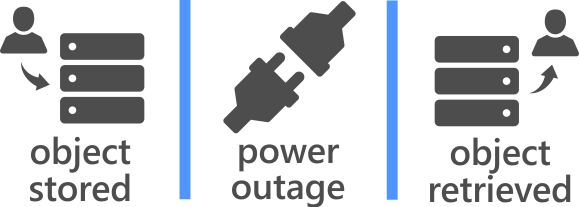
Gotovo sve aplikacija imaju ovakvu potrebu, tj. da objekte koji su kreirani sačuvaju za kasniju upotrebu. Razlozi za ovako nešto su brojni. Aplikacija može baratati podacima o zaposlenima, konfiguracionim podacima aplikacije, podacima o statistici korišćenja, te se može reći da su potrebe praktično neograničene.

Aplikacija za upravljanje ljudskim resursima, imaće potrebu za čuvanjem podataka o zaposlenima. Svakako, ovakva aplikacija sadržaće mnoštvo objekata zaposlenih, a neki tipični podaci za čuvanje mogu biti ime, prezime, radna pozicija, visina dohotka i slično. Sve ove podatke iz objekata zaposlenih potrebno je sačuvati na neku vrstu perzistentne memorije. U toku izvršavanja aplikacije, ovi podaci postoje unutar [radne memorije](https://www.link-elearning.com/linkdl/opisPojma.php?id=142474) koja je takozvana *transient* ili *volatile* memorija, pa se može reći da su takvi podaci samo privremeno sačuvani. U slučaju nestanka napajanja, ovakve memorije nisu u mogućnosti da sačuvaju svoje stanje, tako da dolazi do gubitka podataka, što je dočarano slikom 1.1.



*Slika 1.1 – Tranzijentan objekat*

Sa druge strane, postoje memorije koje mogu da čuvaju podatke čak i nakon gubitka napajanja, i to su takozvane [*non-volatile* memorije](https://www.link-elearning.com/linkdl/opisPojma.php?id=142475). Perzistencija jednog objekta, korišćenjem takve vrste memorije, prikazana je slikom 1.2.

****

*Slika 1.2 – Perzistentan objekat*

Na kraju se može zaključiti da je osnovna uloga perzistencije čuvanje podataka, tako da oni nadžive izvršavanje aplikacije.

**Postizanje perzistencije**

Perzistencija podataka se može postići na mnoge načine. Podaci se mogu sačuvati u fajlu, i to u obliku najobičnijeg teksta. Tekstualni format u kome se podaci najčešće predstavljaju jeste [XML](https://www.link-elearning.com/linkdl/opisPojma.php?id=142476) ili [JSON](https://www.link-elearning.com/linkdl/opisPojma.php?id=142477). Sledeći primer ilustruje kako je moguće izvršiti serijalizaciju objekta korišćenjem XML-a. Objekat koji će biti serijalizovan biće tipa Person. Ovaj tip imaće sledeću strukturu:

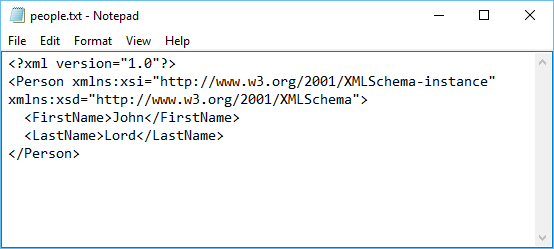
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | [Serializable]          public class Person          {              public string FirstName { get; set; }              public string LastName { get; set; }              public Person(string firstname, string lastname)              {                  FirstName = firstname;                  LastName = lastname;              }              public Person()              {              }          } |

|  |
| --- |
| ***Napomena:****klase objekata za koje će se javiti potreba za serijalizacijom, neophodno je da budu dekorisane [Serializable] anotacijom. Ova anotacija se realizuje klasom  SerializableAttribute, a ukazuje na to da instance određene klase mogu da budu serijalizovane. Okruženje emituje izuzetak tipa SerializationException, ukoliko se serijalizacija pokuša izvršiti nad bilo kojim objektom čiji tip nije dekorisan ovim atributom.* |

Kod za obavljanje serijalizacije jednog objekta prikazane klase Person u tekstualni XML oblik je sledeći:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | Person person = new Person("John", "Lord");              XmlDocument xmlDocument = new XmlDocument();              XmlSerializer serializer = new XmlSerializer(person.GetType());              using (MemoryStream stream = new MemoryStream())              {                  serializer.Serialize(stream, person);                  stream.Position = 0;                  xmlDocument.Load(stream);                  xmlDocument.Save("people.txt");                  stream.Close();              } |

Nakon izvršavanja prikazanog koda u folderu sa fajlovima izvornog koda naći će se i fajl sa nazivom *people.txt,* sa sadržajem kao na slici 1.3.



*Sika 1.3 – Izgled fajla koji čuva objekat serijalizovan u XML formatu*

Da bi se izvršio obrnuti proces, tj. da bi se iz XML fajla pročitali podaci i pretvorili u objekte, dovoljno je uraditi sledeće:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | Person person = new Person();              XmlDocument xmlDocument = new XmlDocument();              XmlSerializer serializer = new XmlSerializer(person.GetType());              StreamReader reader = new StreamReader("people.txt");              person = (Person)serializer.Deserialize(reader);              reader.Close();              Console.WriteLine(person.FirstName + " " + person.LastName); |

|  |
| --- |
| ***Napomena:****prikazani primeri imaju ulogu samo da ilustruju jedan od osnovnih načina za postizanje perzistencije, i njihov cilj nikako nije da zalaze u analizu XML serijalizacije.* |

Pored tekstualnog oblika za čuvanje podataka, današnje aplikacije se uglavnom za postizanje perzistencije oslanjaju na bazu podataka.

**Da li se čuvanje podataka u radnoj memoriji računara smatra perzistencijom?**

 Da

 Ne

**Šta je baza podataka?**

Priča o podacima ne može da prođe bez definisanja dobro poznatog pojma *baze podataka*. Najjednostavnije rečeno, baza podataka je kolekcija strukturiranih informacija. Osnovna uloga baze je da upravlja velikom količinom podataka i da na organizovan i strukturiran način omogući korisniku rukovanje podacima.

Sistem za upravljanje bazom podataka (*database management system* ili skraćeno DBMS) je program koji omogućava korisniku kreiranje i održavanje baza podataka, kao i pisanje naredbi za čitanje, modifikovanje, brisanje i unos podataka.

Sistem za upravljanje [relacionom bazom podataka](https://www.link-elearning.com/linkdl/opisPojma.php?id=142459) naziva se ***R****elational****D****ata****B****ase****M****anagement****S****ystem*ili skraćeno RDBMS. Postoji nekoliko vodećih sistema za upravljanje bazama podataka, od kojih su najpoznatiji [Microsoft Sql Server](https://www.link-elearning.com/linkdl/opisPojma.php?id=142460), [MySQL](https://www.link-elearning.com/linkdl/opisPojma.php?id=142461), [PostgreSQL](https://www.link-elearning.com/linkdl/opisPojma.php?id=142462) i [Oracle Database 12c](https://www.link-elearning.com/linkdl/opisPojma.php?id=142463). U kursu će pažnja biti posvećena Microsoft SQL Server sistemu, iako će obrađene tehnologije biti primenjive i na druge sisteme za upravljanje podacima.

**ADO.NET**

Nakon uvodne priče o pojmu perzistencije i o osnovnim metodama za njeno postizanje, biće definisane tehnologije koje je Microsoft razvio primarno, kako bi programerima olakšao upravljanje podacima koji se nalaze u bazi podataka.

Pre .NET platforme programeri su za pristup podacima koristili tehnologije kao što su [ODBC](https://www.link-elearning.com/linkdl/opisPojma.php?id=142464), [OLE DB](https://www.link-elearning.com/linkdl/opisPojma.php?id=142465) i [ActiveX Data Object](https://www.link-elearning.com/linkdl/opisPojma.php?id=142466) (ADO). Sa predstavljanjem .NET platforme, Microsoft je kreirao nov način za rad sa podacima, nazvan ADO.NET.

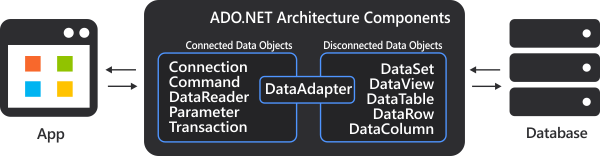
ADO.NET je set klasa koje izlažu servise za pristup podacima .NET programerima, na taj način obezbeđujući bogat set komonenti za kreiranje distribuiranih i sadržajno bogatih aplikacija. ADO.NET je integralni deo .NET okvira i omogućava pristup podacima u relacionom, XML ili aplikativnom obliku.

ADO.NET je kreiran kako bi podržao diskonektovan pristup podacima, tesnu integraciju sa XML-om i mogućnost kombinovanja podataka iz više izvora, a sve to na osnovi .NET platforme.

**ADO.NET arhitektura**

ADO.NET poseduje dva tipa arhitekturinih komonenti za izgradnju aplikacija koje rukuju podacima. Takve komponente omogućavaju rad u konektovanom i diskonetkovanom okruženju. Unutar .NET okvira ADO.NET funkcionalnosti se nalaze u prostoru imena System.Data tako se da se klase, koje omogućavaju i konektovani i diskonektovani pristup, nalaze u istom prostoru imena.

 Slika 1.4 prikazuje osnovnu arhitekturu ADO.NET tehnologije.



*Slika 1.4 – ADO.NET arhitektura*

Sa slike 1.4 se može lako zaključiti da ADO.NET arhitektura poseduje dva skupa komonenti. Tako su pojmovi Connection, Command, DataReader i ostali, karakteristični uglavnom za konektovani pristup. Neki od tipova karakterističnih za diskonektovani pristup su DataSet, DataView, DataTable i tako dalje. Ali, koja je razlika između ova dva pristupa?

***Konektovano okruženje***

Prilikom korišćenja konektovanog okruženja bilo kakva manipulacija podacima zahteva postojanje otvorene konekcije sa bazom. Zbog toga se ovakav pristup i naziva konektovani pristup. Konekcija se otvara neposredno pre izvršavanja komande, a zatvara odmah nakon završenog posla. Ovo je, inače, osnovni pristup za rukovanje podacima.

Osnovni objekti koji se koriste prilikom realizacije ovakvog pristupa su Connection, Command, DataReader i ostali. Connection objekat omogućava uspostavljanje veze sa bazom podataka, a Command objekat omogućava kreiranje komandi za čitanje i modifikovanje podataka, izvršavanje uskladištenih rutina, i slanje i prijam informacija o parametrima.

Objekat klase DataReader je jedna od najznačajnijih komonenti konektovanog okruženja. DataReader omogućava visokoefikasno čitanje podataka u obliku toka. Na kraju, DataAdapter je komponenta koja predstavlja most između DataSet objekta i izvora podataka.

|  |
| --- |
| ***Napomena:****rukovanje podacima u konektovanom okruženju je predmet naredne lekcije pod nazivom Rukovanje podacima u povezanom okruženju.* |

***Diskonektovano okruženje***

Osnovna komponenta diskonektovanog okruženje jeste tip DataSet. Ova komponenta ADO.NET okvira je specijalno dizajnirana za pristup podacima nezavisno od tipa skladišta. Kao rezultat ove osobine, DataSet se može koristiti sa većim brojem različitih skladišta. DataSet svoje funkcionisanje zasniva na kolekciji DataTable objekata, koji se sastoje iz redova i kolona koji predstavljaju podatke.

|  |
| --- |
| ***Napomena:****u lekciji Rukovanje podacima u nepovezanom okruženju, koja se nalazi u ovom modulu, više će biti reči o radu sa podacima i to korišćenjem nepovezanog okruženja.* |

***Koje okruženje koristiti?***

Jedno od značajnih pitanja odnosi se na situacije u kojima je dobro koristiti neko od navedenih okruženja. Diskonektovano okruženje je dobro koristiti u sledećim situacijama:

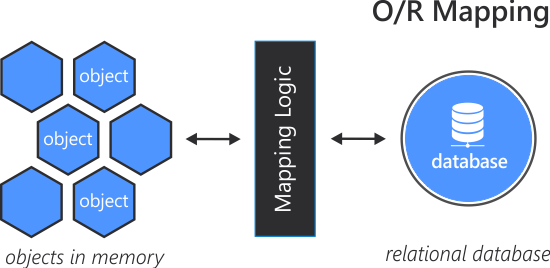
* + kada je potrebno podatke keširati lokalno; ukoliko je potrebno samo pročitati traženi podatak ili podatke, najbolje je koristiti DataReader;
  + kada je potrebno podacima rukovati dinamički;
  + kada je potrebno obavljati zahtevno procesiranje podataka, koje bi inače držalo konekciju otvorenu značajan vremenski period.

Ukoliko unutar aplikacije nisu potrebne upravo iznete osobine diskonektovanog okruženja, korišćenjem konektovanog okruženja se mogu postići znatna poboljšanja performansi. Razlog je naravno taj što konektovani pristup koristi DataReader objekat koji omogućava isključivo čitanje, i to samo u jednom smeru (unapred), te na taj način predstavlja veoma brzo rešenje za čitanje podataka. Iako i DataAdapter u pozadini koristi DataReader za popunjavanje DataSet objekta, korišćenjem DataReadera ostvaruje se napredak u performansama, tako što se štedi memorija koja bi inače bila potrošena na čuvanje podataka DataSeta.

**Objektno-relaciono mapiranje**

Pored upoznavanja sa osnovama ADO.NET tehnologije, veći deo ovog kursa biće posvećen pojmu objektno-relacionog mapiranja, korišćenjem tehnologije ADO.NET Entity Framework.

Pojam *objektno-relaciono mapiranje* se odnosi na programske tehnike koje obavljaju konvertovanje podataka između nekompatibilnih sistema za predstavljanje podataka. Objektno-relaciono mapiranje ili skraćeno ORM predstavlja mehanizam koji omogućava kreiranje mapiranja između podataka i objektno orijentisanih entiteta softverske arhitekture. Drugim rečima, objektno-relaciono mapiranje omogućava programerima da podacima rukuju korišćenjem objekata i klasa, a takav proces ilustrovan je slikom 1.5.



*Slika 1.5 – Objektno-relaciono mapiranje*

**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, cload i service aplikacije.

Entity Framework je primarno alat za vršenje objektno-relacionog mapiranja. 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.

|  |
| --- |
| ***Napomena:****najveći deo ovog kursa biće posvećen korišćenju Entity Framework tehnologije. U nastavku pojam Entity Framework će često biti referisan korišćenjem skraćenice EF.* |

**LINQ**

Značajni deo ovog kursa obuhvatiće i korišćenje tehnologije pod nazivom LINQ. LINQ omogućava standardni, objedinjeni pristup za pretragu i modifikovanje podataka koji može da podrži praktično bilo koji tip podataka. Tako LINQ dozvoljava pretragu različitih tipova podataka, uključujući relacione baze, XML dokumenta, pa čak i strukture podataka smeštene u radnoj memoriji.

U nastavku ovog kursa LINQ tehnologija će najviše biti korišćena za formiranje upita, a prilikom rada sa podacima korišćenjem EF-a.

|  |
| --- |
| ***Napomena:****LINQ tehnologiji će biti posvećen kompletan modul ovoga kursa pod nazivom LINQ.* |

**Korišćenje primera u ovom kursu**

U svim narednim lekcijama ovog kursa, tehnike koje budu teoretski predstavljane, biće potkrepljene praktičnim primerima. Kako bi ti praktični primeri mogli efikasno da se prate, neophodno je na lokalnoj mašini imati aktiviran *MsSQL server* i bazu podataka *Northwind*. Northwind je jedna od test baza koju je razvio Microsoft. Pored Northwind baze podataka, postoji i AdventureWorks test baza podataka koja je novijeg datuma. Ipak, zbog jednostavnosti u ovom kursu će biti korišćena Northwind baza.

Northwind baza se može preuzeti sa sledeće web adrese:

<https://northwinddatabase.codeplex.com/>

Na navedenoj URL adresi nalazi se i dijagram tabela tako da je moguće upoznati se i sa osnovnom strukturom ove baze.

Baza se na lokalni računar preuzima u obliku fajla sa ekstenzijom bak. Ovo je fajl koji se koristi za pravljenje rezervne kopije podataka. Postoji nekoliko načina na koji se ovaj fajl može iskoristiti za uspostavljanje Northwind baze na lokalnom računaru. Najjednostavniji podrazumeva korišćenje [Microsoft SQL Server Management Studioa](https://www.link-elearning.com/linkdl/opisPojma.php?id=142467). Moguće je restore baze podataka obaviti i korišćenjem razvojnog okruženja Visual Studio i Server Explorer prozora. Server Explorer prozor unutar Visual Studio razvojnog okruženja omogućava izvršavanje osnovnih operacija nad bazama na serveru, ali ne podržava automatizovani restore baze korišćenjem .bak fajla, tako da je za obavljanje takvog posla neophodno napisati restore SQL upit. Ovaj upit je moguće uputiti i korišćenjem Command Prompta (slika 1.6 ), upotrebom alata [*sqlcmd*](https://www.link-elearning.com/linkdl/opisPojma.php?id=142468).

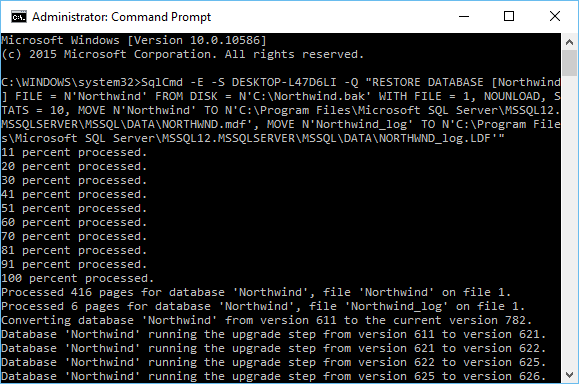
https://www.link-elearning.com/linkdl/coursefiles/1142/MDA_01_06.jpg

*Slika 1.6 – Command Prompt*

Dovoljno je otvoriti Command Prompt i uneti sledeću komandu:

SqlCmd -E -S DESKTOP-L47D6LI -Q "RESTORE DATABASE [Northwind] FILE = N'Northwind' FROM DISK = N'C:\Northwind.bak' WITH FILE = 1, NOUNLOAD, STATS = 10, MOVE N'Northwind' TO N'C:\Program Files\Microsoft SQL Server\MSSQL12.MSSQLSERVER\MSSQL\DATA\NORTHWND.mdf', MOVE N'Northwind\_log' TO N'C:\Program Files\Microsoft SQL Server\MSSQL12.MSSQLSERVER\MSSQL\DATA\NORTHWND\_log.LDF'"

Efekat navedene komande (slika 1.7 ) jeste kreiranje baze podataka na lokalnom SQL serveru.



*Slika 1.7 – Efekat izvršavanja RESTORE naredbe korišćenjem CMD-a*

Da bi navedena komanda funkcionisala, neophodno je ispravno navesti vrednosti nekoliko parametara. Prvo je potrebno navesti naziv lokalnog servera. U primeru je to *DESKTOP-L47D6LI*. Kod Vas će ta vrednost naravno biti drugačija, a ukoliko je ne znate dovoljno je za naziv postaviti tačku (.), što će predstavljati lokalni računar. Takođe, veoma bitan detalj za uspešno izvršenje navedene komande jeste i putanja na kojoj se nalazi fajl sa bak ekstenzijom. Tu je neophodno postaviti putanju do fajla na Vašem fajl sistemu. Deo naredbe u kome se podešava takva putanja je sledeći:

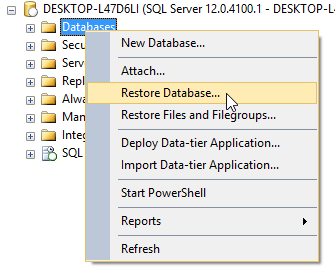
FROM DISK = N'C:\Northwind.bak'

Takođe, putanja na koju je potrebno obratiti pažnju je i ona kojom se definiše folder u kome će se nalaziti stvarni, fizički podaci. Ova putanja zavisi od verzije SQL servera, a u primeru je sledeća:

C:\Program Files\Microsoft SQL Server\MSSQL12.MSSQLSERVER\MSSQL\DATA\

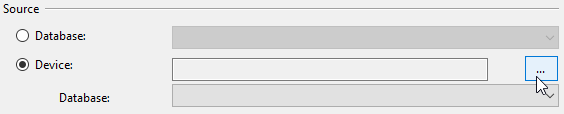
Potrebno je da na Vašim mašinama unutar foldera Program Files i instalacionog foldera Ms SQL Server potražite DATA folder.

Nakon uspešno izvršene komande baza Northwind će biti kreirana na lokalnom SQL serveru. Ukoliko Vam se navedeni postupak čini komplikovanim, postoji i znatno jednostavnije rešenje koje podrazumeva korišćenje Microsoft SQL Server Management Studio razvojnog okruženja. Za tako nešto, dovoljno je odabrati opciju *Restore Database...* iz kontekstnog menija koji se dobija desnim klikom na folder *Databases*, kao na slici 1.8.



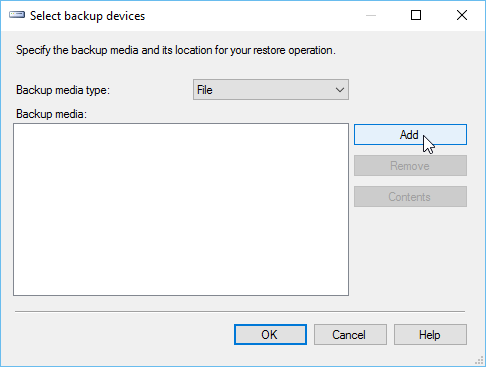
*Slika 1.8 – Restore Database opcija*

U prozoru koji se zatim otvara, potrebno je čekirati opciju Device i pritisnuti dugme, kao na slici 1.9.



*Slika 1.9 – Restore Database opcija*

Navedeno kao efekat ima otvaranje prozora unutar kojeg je potrebno izvršiti dodavanje bak fajla baze podataka. Za to je dovoljno odabrati dugme Add, kao na slici 1.10.

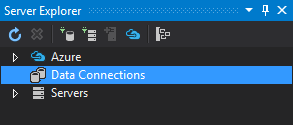


*Slika 1.10 – Dodavanje novog bak fajla*

U prozoru koji se otvorio, treba pronaći bak fajl baze podataka koju je potrebno vratiti na server i odabrati Ok, pa zatim opet Ok. U inicijalnom prozoru je takođe neophodno odabrati Ok i Restore baze podataka će biti obavljen.

**Konektovanje na bazu korišćenjem Visual Studio razvojnog okruženja**

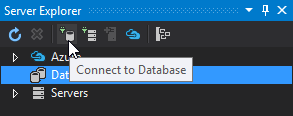
Kako bi se konekcija sa bazom podataka ostvarila korišćenjem Visual Studio razvojnog okruženja, moguće je koristiti Server Explorer prozor, prikazan slikom 1.11.



*Slika 1.11 – Server Explorer*

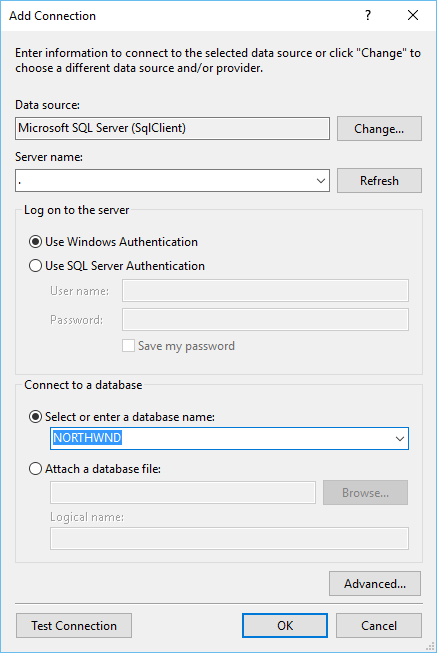
Za aktiviranje prozora Server Explorer dovoljno je odabrati opciju *View -> Server Explorer*.

Da bi se kreirala nova konekcija korišćenjem Visual Studio razvojnog okruženja, potrebno je kliknuti na opciju *Connect to Database*, kao na slici 1.12.



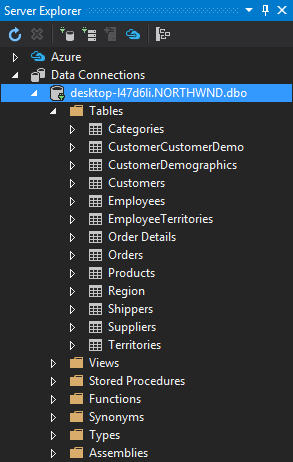
*Slika 1.12 – Connect to Database opcija*

U prozoru koji se zatim otvara potrebno je konfigurisati novu konekciju, kao na slici 1.13.



*Slika 1.13 – Konfigurisanje konekcije*

Za naziv servera moguće je uneti karakter tačka, koji se odnosi na lokalni server. U okviru regiona *Connect to a database* potrebno je selektovati opciju *Select or enter a database name*, i iz padajućeg menija odabrati bazu sa kojom je potrebno ostvariti konekciju. U primeru to je baza *Northwind*. Nakon definisanja osobina konekcije potrebno je odabrati dugme Ok, a nakon dodavanja nove konekcije, u okviru Server Explorer prozora, moguće je videti strukturu kompletne baze podataka sa kojom je ostvarena konekcija, kao na slici 1.14.



*Slika 1.14 – Struktura baze Northwind u Server Explorer prozoru*

Folder *Tables* sadrži sve tabele odabrane baze podataka. Folder *Views* sadrži poglede baze, folder *Store Procedures* uskladištene procedure itd. Sa ovim različitim objektima baze podataka Northwind upoznaćemo se tokom trajanja kursa.