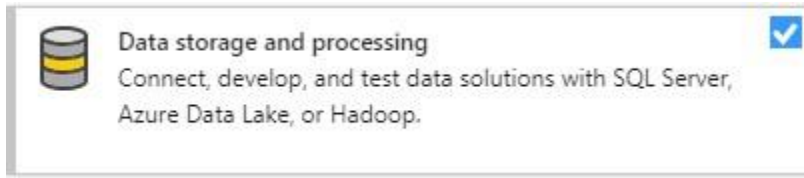


Database

1.mora install



2.

Tools

Connect to Database

Enter Server Name and database vo Advance ima connection string

Server Explorer

DataConnections

ima connection string

ORM se EntityFramework,Dapper

Entity Framework is built on ADO.NET and it uses ADO.NET inside.

ADO.NET

```
string connectionString = "Data Source=ALEK;Initial Catalog=northwind;Integrated Security=True";

using (SqlConnection connection = new SqlConnection(connectionString))
{
    string queryString = "select * from Contacts";
    SqlCommand command = new SqlCommand(queryString, connection);

    try
    {
        connection.Open();

        SqlDataReader reader = command.ExecuteReader();
        reader[0] [1] [2] se vrednosti od kolonite ili moze reader["imeNaKolona"]
        while (reader.Read())//Read e za next record a //NextResult() za
        next result(poveke tabeli)
        {
            Debug.WriteLine("reader0" + reader[0].ToString());
            Debug.WriteLine("reader1" + reader[1]);
            Debug.WriteLine("reader2" + reader[2]);
        }
        reader.Close();
        connection.Close();
    }
    catch (Exception e)
    {
        Debug.WriteLine(e.ToString());
    }
}
```

Kako da izvrsam query

LINQ

```
IEnumerable<Student> QuerySyntax = DbSet.Where(x => x.Status == 1)
```

```
IQueryable<Student> QuerySyntax = DbSet.Where(x => x.Status == 1)
```

ADO.NET

```
string connectionString =
    "Data Source=(local);Initial Catalog=Northwind;"
    + "Integrated Security=true";

// Provide the query string with a parameter placeholder.
string queryString =
    "SELECT ProductID, UnitPrice, ProductName from dbo.products "
    + "WHERE UnitPrice > @pricePoint "
    + "ORDER BY UnitPrice DESC;";

// Specify the parameter value.
int paramValue = 5;

// Create and open the connection in a using block. This
// ensures that all resources will be closed and disposed
// when the code exits.
using (SqlConnection connection =
    new SqlConnection(connectionString))
{
    // Create the Command and Parameter objects.
    SqlCommand command = new SqlCommand(queryString, connection);
    command.Parameters.AddWithValue("@pricePoint", paramValue);

    // Open the connection in a try/catch block.
    // Create and execute the DataReader, writing the result
    // set to the console window.
    try
    {
        connection.Open();
        SqlDataReader reader = command.ExecuteReader();
        while (reader.Read())
        {
            Console.WriteLine("\t{0}\t{1}\t{2}",
                reader[0], reader[1], reader[2]);
        }
        reader.Close();
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
    Console.ReadLine();
}
```

Dapper

```
string sql = "SELECT TOP 10 * FROM OrderDetails";

        using (var connection = new SqlConnection(_connectionString)
        {

                var orderDetails =
connection.QueryAsync<OrderDetail>(sql).Result.ToList();

                Console.WriteLine(orderDetails.Count());
        }
```

Proceduri

ADO.NET

```
using (SqlConnection conn = new
SqlConnection("Server=(local);DataBase=Northwind;Integrated Security=SSPI")) {
    conn.Open();

string connectionString = "Data Source=ALEK;Initial Catalog=northwind;Integrated
Security=True";
```

```
        using (SqlConnection connection = new SqlConnection(connectionString))
        {

                SqlCommand cmd = new SqlCommand("[Customers By City]", connection);

                // 2. set the command object so it knows to execute a stored procedure
                cmd.CommandType = CommandType.StoredProcedure;

                // 3. add parameter to command, which will be passed to the stored procedure
                cmd.Parameters.Add(new SqlParameter("@param1", "Berlin"));
ili
command.Parameters.AddWithValue("@param", "Berlin");
                connection.Open();

                // execute the command
                using (SqlDataReader reader = cmd.ExecuteReader())
                {
                        // iterate through results, printing each to console
                        while (reader.Read())
```

```

        {
            Debug.WriteLine("reader0" + reader[0].ToString());
            Debug.WriteLine("reader1" + reader[1]);
            Debug.WriteLine("reader2" + reader[2]);
            Debug.WriteLine("name " + reader["ContactName"]);
        }
    }
}

```

Dapper

var res = await conn.QueryAsync<TEntity> (storedProcedure.ToString(), req, CommandType: CommandType.StoredProcedure);

SqlDataReader is connection oriented and the connection needs to be opened explicitly, by calling the **Open()** method on the connection object, before calling the **ExecuteReader()** method of the command object.

using takes care of closing the connection but you need to handle exceptions

using (SqlConnection connection = new SqlConnection(connectionString))

```

{
}

```

is the same as

```
SqlConnection connection = null;
```

```
try
```

```
{
```

```
    connection = new SqlConnection(connectionString);
```

```
}
```

```
finally
```

```
{
```

```
    if(connection != null)
```

```
        ((IDisposable)connection).Dispose();
```

```
}
```

Common

```
try
```

```
{
```

```
    using (var myObject = new MyClass())
```

```
    {
```

```
        // something here...
```

```
    }
```

```
}
```

```
catch(Exception ex)
```

```
{
```

```
    // Handle exception
```

```
}
```

SqlCommand class is used to prepare an SQL statement or StoredProcedure that we want to execute on a SQL Server database.

The most commonly used methods of the SqlCommand class

1. ExecuteReader - Use when the T-SQL statement returns more than a single value. For example, if the query returns rows of data.

2. ExecuteNonQuery - Use when you want to perform an Insert, Update or Delete operation.

3. ExecuteScalar - Use when the query returns a single(scalar) value. For example, queries that return the total number of rows in a table.

Object reader1 = command1.ExecuteScalar(); ExecuteScalar e poefikasno od ExecuteReaderot ako vrakame eden rezultat.

Object reader1 = command1.ExecuteNonQuery(); //vraka total rows changed

//prevent sql injection attack

string queryString = "select * from Contacts where City like @param";

command.Parameters.AddWithValue("@param", "Berlin");

get output from procedure

```
SqlCommand cmd = new SqlCommand("spAddEmployee", con);
cmd.CommandType = System.Data.CommandType.StoredProcedure;

cmd.Parameters.AddWithValue("@Name", txtEmployeeName.Text);
cmd.Parameters.AddWithValue("@Gender", ddlGender.SelectedValue);
cmd.Parameters.AddWithValue("@Salary", txtSalary.Text);

SqlParameter outputParameter = new SqlParameter();
outputParameter.ParameterName = "@EmployeeId";
outputParameter.SqlDbType = System.Data.SqlDbType.Int;
outputParameter.Direction = System.Data.ParameterDirection.Output;
cmd.Parameters.Add(outputParameter);

con.Open();
cmd.ExecuteNonQuery();
```

SqlDataReader is connection oriented, meaning it requires an active and open connection to the data source.

SqlDataAdapter and **DataSet** provides us with disconnected data access model.

A **SqlDataAdapter** is typically used to fill a **DataSet** or **DataTable** and so you will have access to the data after your connection has been closed.

SqlDataReader is a fast forward-only and connected cursor which tends to be generally quicker than filling a **DataSet/DataTable**

```
string CS = ConfigurationManager.ConnectionStrings["DBCS"].ConnectionString;
using (SqlConnection con = new SqlConnection(CS))
{
    SqlDataAdapter da = new SqlDataAdapter("Select * from tblProductInventory", con);
    DataSet ds = new DataSet();
    da.Fill(ds);
}
|
```

so Fill se otvara konekcija se pravi execute na query se zatvara i se puni dataset
ako ima poveke tabeli vo dataSet so d.Table[0] ili d.Table["Table"] se pristapuvaat

SqlDataAdapter for procedure

```
SqlDataAdapter da = new SqlDataAdapter("spGetProductInventory", con);  
da.SelectCommand.CommandType = CommandType.StoredProcedure;  
DataSet ds = new DataSet();  
da.Fill(ds);
```

with parameters

```
SqlDataAdapter da = new SqlDataAdapter("spGetProductInventoryById", con);  
da.SelectCommand.CommandType = CommandType.StoredProcedure;  
da.SelectCommand.Parameters.AddWithValue("@ProductId", TextBox1.Text);  
  
DataSet ds = new DataSet();  
da.Fill(ds);
```

the dataSet will have table Students filled with rows

```
dataAdapter.Fill(dataSet, "Students");
```

put it in cache Cache["Data"]=ds

get it from cache (DataSet)Cache["Data"]

Disconnected Data Access

You now have data in the DataSet and there is no active connection to the database. At this point you can make any changes(insert, update, delete) to the data in the DataSet. Only the data in the DataSet is changed, the underlying database table data is not changed. To update the underlying database table, invoke SqlDataAdapter.Update() method.

```
dataAdapter.Update(DataSetObject, "TableName");
```

Make sure that UPDATE, DELETE and INSERT commands are associated with SqlDataAdapter object when Update() method is called, otherwise there would be a runtime exception.

```
string strUpdateCommand = "Update tblStudents set Name = @Name, Gender = @Gender, TotalMarks = @TotalMarks w  
SqlCommand updateCommand = new SqlCommand(strUpdateCommand, con);  
updateCommand.Parameters.Add("@Name", SqlDbType.NVarChar, 50, "Name");  
updateCommand.Parameters.Add("@Gender", SqlDbType.NVarChar, 20, "Gender");  
updateCommand.Parameters.Add("@TotalMarks", SqlDbType.Int, 0, "TotalMarks");  
updateCommand.Parameters.Add("@Id", SqlDbType.Int, 0, "Id");
```

```
da.UpdateCommand = updateCommand;
```

```
string strDeleteCommand = "Delete from tblStudents where Id = @Id";  
SqlCommand deleteCommand = new SqlCommand(strDeleteCommand, con);  
deleteCommand.Parameters.Add("@Id", SqlDbType.Int, 0, "Id");  
da.DeleteCommand = deleteCommand;
```

```
da.Update(ds, "Students");
```

```

DataSet ds = (DataSet)Cache["DATASET"];
DataRow newRow = ds.Tables["Students"].NewRow();
newRow["Id"] = 101;
//ds.Tables["Students"].Rows.Add(newRow);

foreach (DataRow dr in ds.Tables["Students"].Rows)
{
    if (dr.RowState == DataRowState.Deleted)
    {
        Response.Write(dr["Id", DataRowVersion.Original].ToString() + " - " + dr.RowState.ToString() + "<br/>");
    }
    else
    {
        Response.Write(dr["Id"].ToString() + " - " + dr.RowState.ToString() + "<br/>");
    }
}

```

dataset has table that contain rows which can be accessed

ORM od query(tabela) vo objecti

```

string connectionString = ConfigurationManager.ConnectionStrings["DBCS"].ConnectionString;
SqlConnection connection = new SqlConnection(connectionString);
string selectQuery = "Select * from tblStudents";
SqlDataAdapter dataAdapter = new SqlDataAdapter(selectQuery, connection);

DataSet dataSet = new DataSet();
dataAdapter.Fill(dataSet, "Students");

GridView1.DataSource = from DataRow in dataSet.Tables["Students"].AsEnumerable()
                        select new Student { ID = Convert.ToInt32(dataRow["Id"]),
                                              Name = dataRow["Name"].ToString(),
                                              Gender = dataRow["Gender"].ToString(),
                                              TotalMarks = Convert.ToInt32(dataRow["TotalMarks"])};

```

context klasa vo koja ke gi ima dbsetovite i ke nasleduva od DbContext kade ke mu prati

connectionString vo base

class Context:DbContext

public Context():base(ConnectionString)

kako da dobijam konekcija

so

string connectionString =

ConfigurationManager.ConnectionStrings["myConnectionString"].ConnectionString;

<connectionStrings>


```
<add name="myConnectionString" connectionString="Data Source=ALEK;Initial
Catalog=northwind;Integrated Security=True" />
</connectionStrings>
```

ili

so override na OnConfiguring metoda koja se naoga na DbContext

```
protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)
{
    optionsBuilder.UseSqlServer(ConnectionString);
}
```

The DataBind() method binds the data to the control on which you have invoked it. If this method is not called, the data will not be bound to the control and will not be displayed. Setting DataSource property and invoking DataBind() method is required if you want the data to be displayed in the databound controls like DropDownList, ListBox, GridView, Repeater etc.