DBMS – Lab 2

Here are a few important notions you should know before starting to work on your assignment:

- **SqlCommand** – a T-SQL statement(SQL query: SELECT, INSERT, UPDATE, DELETE or stored procedure) that you want to execute against a SQL Server Database; takes as arguments the query/stored procedure and the connection to the database

```
private void exampleSqlCommand()
{
    using(SqlConnection conn = new SqlConnection(connectionString)){

        conn.Open();
        SqlCommand cmd = new SqlCommand("SELECT COUNT(*) FROM Dog", conn);
        int count = (int)cmd.ExecuteScalar();
        Console.WriteLine("Dogs Count: " + count);
    }
}
```

- **SqlParameter** – argument to the SQL command

```
private void exampleSqlParameter()
{
    using (SqlConnection conn = new SqlConnection(connectionString))
    {
        int breedId = int.Parse(breedTextBox.Text);

        conn.Open();
        SqlCommand cmd = new SqlCommand("SELECT COUNT(*) FROM Dog WHERE breedId = @breedId", conn);
        cmd.Parameters.AddWithValue("@breedId", breedId);

        int count = (int)cmd.ExecuteScalar();
        Console.WriteLine("Dogs Count: " + count);
    }
}
```

- **DataSet** – in-memory representation of your database structure; it can hold multiple DataTable objects
- **SQLDataAdapter –** a bridge between your database and your **DataSet**;
  - ✓ *.fill()* - loads data into the DataSet – does NOT overwrite existing data
  - ✓ *.clear()* – removes rows from DataTable before adding new data – you can call this method when you want to refresh your data that you work with

```
private void exampleData()
{
    using (SqlConnection conn = new SqlConnection(connectionString))
    {
        conn.Open();
        SqlDataAdapter adapter = new SqlDataAdapter("SELECT * FROM Dog", conn);

        DataSet ds = new DataSet();
        adapter.Fill(ds, "Dog");

        foreach (DataRow row in ds.Tables["Dog"].Rows)
        {
            Console.WriteLine(row["dogName"]);
        }
    }
}
```

Lab 1 rewind:

- created a new C# Windows Forms app that uses ADO.NET to communicate with your SQL Server Database
- added a DataSource and checked the connection to it
- chose 2 tables from your database that are in 1:n relationship (parent-child)

Lab 2 requirements:

1. display all the records in the parent
   ADO.NET components needed: **SqlConnection, SqlCommand, SqlDataAdapter, DataSet**

```
private void loadAllDogs()
{
    using (SqlConnection conn = new SqlConnection(connectionString))
    {

        conn.Open();
        SqlCommand cmd = new SqlCommand("SELECT * FROM Dog", conn);

        SqlDataAdapter daChild = new SqlDataAdapter(cmd);
        DataSet ds = new DataSet();

        daChild.Fill(ds);
        allDogs.DataSource = ds.Tables[0]; // allDogs = the name of the DataGridView
        conn.Close();
    }
}
```

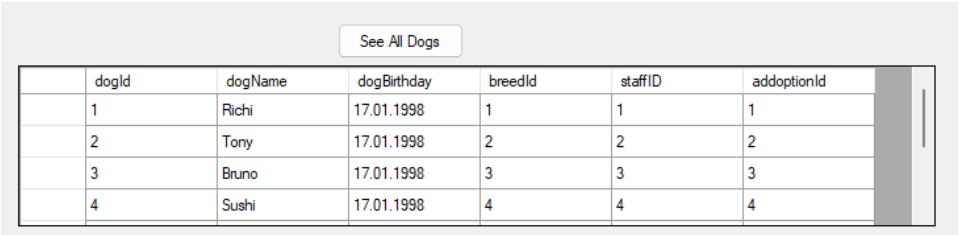
   A. **data is loaded once the application starts** – after you implemented the logic into a method you need to place it into the class constructor so that when the Form will be initialized, your grid will be populated.
   UI components you need: **DataGridView**

```
public dogId()
{
    InitializeComponent();
    loadAllDogs();
}
```

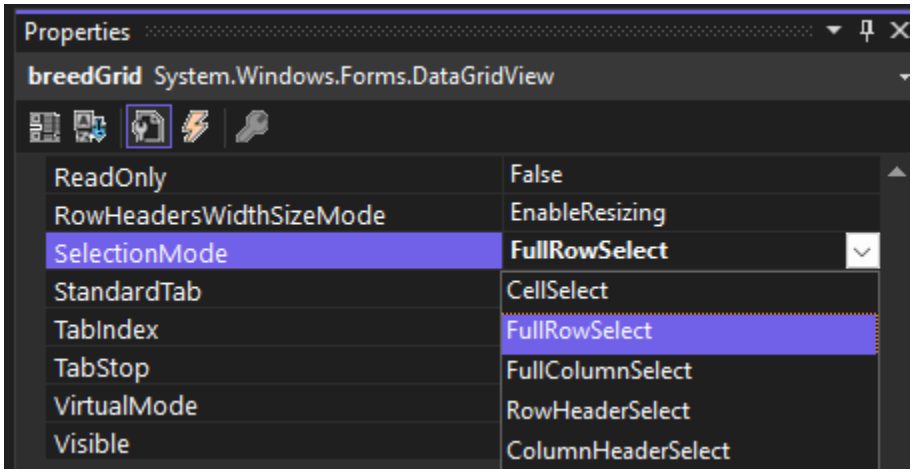
	dogId	dogName	dogBirthday	breedId	staffID	addoptionId
	1	Richi	17.01.1998	1	1	1
	2	Tony	17.01.1998	2	2	2
	3	Bruno	17.01.1998	3	3	3
	4	Sushi	17.01.1998	4	4	4

   B. **data is loaded once a button is hit** - after you implemented the logic into a method you need to place it into the button's click handler. (Hint: double-click on the button and the IDE will take you exactly there)
   UI components you need: **DataGridView + Button**

```
private void seeDogs_Click(object sender, EventArgs e)
{
    loadAllDogs();
}
```



2. display the child records for a specific parent record
   ADO.NET components needed: **SqlConnection, SqlCommand, SqlDataAdapter, DataSet**
   UI components you need: **DataGridView** – you need to make sure that your DataGridView has the **SelectionMode= FullRowSelect** (Hint: right-click on the DataGridView and go to Proprties window in the low right corner of your IDE)



Workflow: once you select a breed in the DataGridView, the dogs of that breed should automatically populate the other DataGridView – this means an event of type **selectionChanged** must be handled, just as we did with the button click above

```
private void breedGrid_selectionChanged(object sender, EventArgs e)
{
    using (SqlConnection conn = new SqlConnection(connectionString))
    {
        conn.Open();

        if (breedGrid.CurrentRow != null)  // Ensure a row is selected
        {
            int breedId = Convert.ToInt32(breedGrid.CurrentRow.Cells["breedId"].Value);
            SqlCommand cmd = new SqlCommand("select * from Dog where breedId=@breedId", conn);
            cmd.Parameters.AddWithValue("@breedId", breedId);

            SqlDataAdapter daChild = new SqlDataAdapter(cmd);
            DataSet dataSet = new DataSet();
            daChild.Fill(dataSet);
            dogsGrid.DataSource = dataSet.Tables[0];
        }
        conn.Close();
    }
}
```

To handle each selection in the breed's DataGridView, we need to "catch" each selection change.

```
public dogId()
{
    InitializeComponent();
    breedGrid.SelectionChanged += breedGrid_selectionChanged;
}
```