ADO.NET Y DATASETS EN C#

ADO.NET

1. es una biblioteca de acceso a datos de Microsoft que proporciona un conjunto de clases y objetos para interactuar con bases de datos relacionales desde una aplicación en C#. Aquí te explico cómo puedes conectarte a una base de datos, realizar consultas y trabajar con parámetros en ADO.NET:

ADO.NET

1. es una biblioteca de acceso a datos de Microsoft que proporciona un conjunto de clases y objetos para interactuar con bases de datos relacionales desde una aplicación en C#. Aquí te explico cómo puedes conectarte a una base de datos, realizar consultas y trabajar con parámetros en ADO.NET:
2. Sin embargo, para trabajar con ADO.NET y DataSets, necesitarás agregar las siguientes directivas using al inicio de tu archivo de código

ejemplo

```
using System.Data;
using System.Data.SqlClient; // O la librería correspondiente al proveedor de base de datos que estés utilizando (por ejemplo, System.Data.OleDb para proveedores de bases de datos OLEDB)
string connectionString = "Data Source=mi_servidor;Initial Catalog=mi_base_de_datos;User ID=mi_usuario;Password=mi_contraseña;";
using (SqlConnection connection = new SqlConnection(connectionString))
{
    connection.Open();
    // Aquí puedes realizar operaciones en la base de datos
}
```

Consultas para cargar una grilla:

- Para cargar una grilla con datos de una base de datos, necesitarás utilizar una consulta SQL. Puedes utilizar un SqlCommand para ejecutar la consulta y un SqlDataAdapter para llenar un DataTable con los resultados.

ejemplo

```
string query = "SELECT * FROM mi_tabla";
using (SqlConnection connection = new SqlConnection(connectionString))
{
    connection.Open();
    using (SqlCommand command = new SqlCommand(query, connection))
    {
        SqlDataAdapter adapter = new SqlDataAdapter(command);
        DataTable dataTable = new DataTable();
        adapter.Fill(dataTable);
        // Asignar el DataTable como origen de datos de la grilla
        miGrilla.DataSource = dataTable;
        miGrilla.DataBind();
    }
}
```

Consultas con parámetros:

- Para realizar consultas con parámetros, puedes utilizar el objeto SqlParameter para definir y asignar valores a los parámetros en la consulta. Esto ayuda a prevenir ataques de inyección de SQL y facilita la reutilización de la consulta con diferentes valores.

ejemplo

```
string query = "SELECT * FROM mi_tabla WHERE nombre = @nombre AND edad > @edad";
using (SqlConnection connection = new SqlConnection(connectionString))
{
    connection.Open();
    using (SqlCommand command = new SqlCommand(query, connection))
    {
        command.Parameters.AddWithValue("@nombre", nombre);
        command.Parameters.AddWithValue("@edad", edad);
        // Ejecutar el comando y trabajar con los resultados
    }
}
```

Consulta para obtener un registro determinado:

- Para obtener un registro específico de la base de datos, puedes utilizar una consulta SQL con una cláusula WHERE que filtre los resultados. Puedes utilizar un SqlDataReader para leer el registro obtenido.

ejemplo

```
string query = "SELECT * FROM mi_tabla WHERE id = @id";
using (SqlConnection connection = new SqlConnection(connectionString))
{
    connection.Open();
    using (SqlCommand command = new SqlCommand(query, connection))
    {
        command.Parameters.AddWithValue("@id", miId);
        using (SqlDataReader reader = command.ExecuteReader())
        {
            if (reader.Read())
            {
                // Leer los valores del registro
                string nombre = reader.GetString(1);
                int edad = reader.GetInt32(2);
                // Realizar las acciones necesarias con los valores obtenidos
            }
        }
    }
}
```

Inserción de datos:

- Para insertar datos en una tabla, debes construir una consulta INSERT INTO y ejecutarla utilizando un objeto SqlCommand. Puedes utilizar parámetros para pasar los valores de los datos que deseas insertar.

ejemplo

```
string query = "INSERT INTO mi_tabla (columna1, columna2) VALUES (@valor1, @valor2)";
using (SqlConnection connection = new SqlConnection(connectionString))
{
    connection.Open();
    using (SqlCommand command = new SqlCommand(query, connection))
    {
        command.Parameters.AddWithValue("@valor1", valor1);
        command.Parameters.AddWithValue("@valor2", valor2);
        command.ExecuteNonQuery();
    }
}
```

Inserción de datos:

- Para insertar datos en una tabla, debes construir una consulta INSERT INTO y ejecutarla utilizando un objeto SqlCommand. Puedes utilizar parámetros para pasar los valores de los datos que deseas insertar.

ejemplo

```
string query = "INSERT INTO mi_tabla (columna1, columna2) VALUES (@valor1, @valor2)";
using (SqlConnection connection = new SqlConnection(connectionString))
{
    connection.Open();
    using (SqlCommand command = new SqlCommand(query, connection))
    {
        command.Parameters.AddWithValue("@valor1", valor1);
        command.Parameters.AddWithValue("@valor2", valor2);
        command.ExecuteNonQuery();
    }
}
```

Dataset tipados

Crear un DataSet tipado:

- Se creara un nuevo origen de datos, incorporando la base de datos y sus objetos deseados.

Llenar el DataSet tipado con datos:

- Después de crear un DataSet tipado, puedes llenarlo con datos provenientes de una base de datos utilizando un SqlDataAdapter y su método Fill.

ejemplo

```
using (SqlConnection connection = new SqlConnection(connectionString))
{
    connection.Open();
    string query = "SELECT * FROM mi_tabla";
    using (SqlCommand command = new SqlCommand(query, connection))
    {
        using (SqlDataAdapter adapter = new SqlDataAdapter(command))
        {
            MiDataSetTipado dataSet = new MiDataSetTipado(); // Reemplaza "MiDataSetTipado" con el nombre de tu DataSet tipado
            adapter.Fill(dataSet.NombreTabla); // Reemplaza "NombreTabla" con el nombre de tu tabla en el DataSet tipado
        }
    }
}
```

Manipular y acceder a los datos en el DataSet tipado:

- Una vez que tienes el DataSet tipado lleno de datos, puedes manipular y acceder a ellos utilizando las propiedades generadas por el diseñador o escritas manualmente en la clase del DataSet tipado. Puedes realizar operaciones como agregar, modificar o eliminar registros en las tablas, y acceder a los valores de las columnas utilizando la sintaxis del DataSet tipado.

ejemplo

```
foreach (MiDataSetTipado.NombreTablaRow row in dataSet.NombreTabla) // Reemplaza "MiDataSetTipado" y "NombreTabla"
con los nombres correspondientes
{
    string columna1 = row.Columna1;
    int columna2 = row.Columna2;
    // Realizar acciones con los datos
}
```

Guardar los cambios en la base de datos:

- Si realizas modificaciones en el DataSet tipado, como agregar, modificar o eliminar registros, puedes utilizar un SqlDataAdapter y su método Update para guardar los cambios en la base de datos.

ejemplo

```
using (SqlConnection connection = new SqlConnection(connectionString))
{
    connection.Open();
    using (SqlDataAdapter adapter = new SqlDataAdapter())
    {
        adapter.Update(dataSet.NombreTabla); // Reemplaza "NombreTabla" con el nombre de tu tabla en el DataSet tipado
    }
}
```

Obteniendo un objeto a partir del Datagrid

```
// Obtener la fila seleccionada en el DataGridView
DataGridViewRow filaSeleccionada = miDataGridView.CurrentRow;

// Verificar si hay una fila seleccionada
if (filaSeleccionada != null)
{
    // Obtener el objeto "Persona" de la fila seleccionada
    Persona personaSeleccionada = filaSeleccionada.DataBoundItem as Persona;

    // Verificar si se pudo obtener el objeto "Persona"
    if (personaSeleccionada != null)
    {
        // Poblar los TextBox con los datos de la persona seleccionada
        txtNombre.Text = personaSeleccionada.Nombre;
        txtEdad.Text = personaSeleccionada.Edad.ToString();
        // Otros campos de la ficha de edición

        // Realizar las acciones necesarias con los datos
    }
}
```

Preguntas?

Credits

Special thanks to all the people who made and released these awesome resources for free:

- Presentation template by SlidesCarnival
- Photographs by Unsplash