Expresiones Lambda y DataGrid

Simplificación de consultas y actualizaciones de datos en C# con sintaxis LINQ y Lambda

¿Qué es una expresión Lambda en C#?

- Una expresión lambda es una forma corta de escribir una función anónima, un fragmento de código que se puede pasar como parámetro a una consulta LINQ u otro método. Utiliza el operador de flecha =>, que significa "va a".
- Sintaxis básica:

```
(parameter) => expression
(parameter) => { statements }
```

¿Por qué es útil?

- Las expresiones lambda hacen que las consultas **LINQ sean más cortas, claras y expresivas**, especialmente al filtrar, ordenar o transformar datos.
- También son la base de los métodos LINQ como:
  - **Where()** -> filtra datos
  - **Select()** -> define que valores o que partes de los datos de origen se van a devolver
  - **OrderBy()** -> ordena datos
  - **FirstOrDefault()** -> recupera un solo elemento

Ejemplo de comparación: sintaxis de consulta frente a sintaxis de Lambda

Operation	LINQ (Query Syntax)	LINQ (Lambda Syntax)
Filter (Age > 24)	`from u in users where u.Age > 24 select u;`	`users.Where(u => u.Age > 24);`
Sort by name	`from u in users orderby u.Name select u;`	`users.OrderBy(u => u.Name);`
Select field	`from u in users select u.Name;`	`users.Select(u => u.Name);`
First element	`(from u in users where u.Age == 30 select u).FirstOrDefault();`	`users.FirstOrDefault(u => u.Age == 30);`
Count	`(from u in users where u.Age >= 18 select u).Count();`	`users.Count(u => u.Age >= 18);`

Ejemplo 1: LINQ (Query Syntax)

```
List<User> users = new List<User>
{
    new User { Name = "Alice", Age = 25 },
    new User { Name = "Bob", Age = 30 },
    new User { Name = "Charlie", Age = 22 }
};

// Query syntax
var filteredUsers = from u in users
                    where u.Age > 24
                    select u;

UserDataGrid.ItemsSource = filteredUsers.ToList();
```

Ejemplo 1: LINQ (Query Syntax)

```
List<User> users = new List<User>
{
    new User { Name = "Alice", Age = 25 },
    new User { Name = "Bob", Age = 30 },
    new User { Name = "Charlie", Age = 22 }
};

// Query syntax
var filteredUsers = from u in users
                    where u.Age > 24
                    select u;

UserDataGrid.ItemsSource = filteredUsers.ToList();
```

The DataGrid shows only users older than 24

Ejemplo 2: LINQ con expresiones Lambda

```
List<User> users = new List<User>
{
    new User { Name = "Alice", Age = 25 },
    new User { Name = "Bob", Age = 30 },
    new User { Name = "Charlie", Age = 22 }
};

// Lambda syntax
var filteredUsers = users
    .Where(u => u.Age > 24)
    .ToList();

UserDataGrid.ItemsSource = filteredUsers;
```

El resultado es el mismo, pero con un código más simple y moderno.

Ejemplo 3: LINQ + Lambda en tu proyecto

```
lstPersonas.Where(p => p.Nombre == txtNombre.Text && p.Apellidos == txtApellidos.Text)
           .ToList()
           .ForEach(p =>
           {
               p.Nombre = txtNombre.Text;
               p.Apellidos = txtApellidos.Text;
               p.Edad = int.Parse(txtEdad.Text);
           });
```

Explicación:

- **Where(…)** -> filtra la lista buscando personas con el mismo nombre y apellido que los que están en los TexBox.
- **ToList()** -> crea una lista en memoria con los resultados del Where anterior para poder usar ForEach.
- **ForEach(…)** -> recorre la lista creada por el ToList anterior actualizando las propiedades de cada persona encontrada.

Ejemplo 3: LINQ + Lambda en tu proyecto

Esto hecho con LINQ sería así:

```
var personasFiltradas =
      from p in lstPersonas
      where p.Nombre == "Antonio" && p.Apellidos == "Sánchez"
      select p;

foreach (var p in personasFiltradas)
{
      p.Nombre = "Tony";
      p.Apellidos = "Redondo";
}

dgvPersonas.ItemsSource = lstPersonas;
```

Resumen 1

- LINQ nos permite consultar datos fácilmente, como SQL, pero dentro de C#.
- Las lambdas hacen que esas consultas sean más cortas y flexibles.
- Juntos, nos permiten filtrar, buscar y actualizar las listas que se muestran en un DataGrid y, más tarde, los datos de una base de datos real

Resumen 2

- **Expresiones lambda:**

Las funciones cortas en línea → hacen que el código sea más legible.

- **LINQ + Lambda:**

Consulta y filtrado de colecciones de manera eficiente.

- **Cuadrícula de datos:**

Muestra visualmente los resultados de la consulta.

Conexión a bases de datos

- Cuando se usa **Entity Framework** o **LINQ to SQL**, se aplica la misma sintaxis lambda, pero en tablas de base de datos en lugar de listas en memoria.
- Ejemplo:

```
var adults = db.Users.Where(u => u.Age >= 18).ToList();
UserDataGrid.ItemsSource = adults;
```

La misma idea funciona: solo cambia la fuente de datos:

Lista en memoria → Tabla de base de datos