	C# TIPS: Lambda Expression	
		Rosa María Zapata Calle Angel Quero Blanquet

## Expresiones Lambda:

Las expresiones lambda en C# son funciones anónimas que permiten definir operaciones de manera concisa. Tienen la sintaxis (parametros) => expresión. Son útiles para escribir funciones cortas y directas, especialmente como argumentos de métodos que aceptan delegados.

Ejemplo:

`x => x * x`

Esto define una función que toma un valor x y devuelve su cuadrado.

## LINQ (Language Integrated Query):

LINQ es una herramienta que permite realizar consultas a colecciones (como listas, arrays, colecciones o bases de datos) de manera similar a SQL, pero directamente en código. Utiliza métodos como Where, Select, OrderBy, que suelen combinarse con expresiones lambda.

Ejemplo:

`var pares = numeros.Where(n => n % 2 == 0);`

Esto filtra una lista de números, seleccionando solo los pares.

La relación entre las expresiones lambda y LINQ es que las lambda son claves para definir filtros y transformaciones en las consultas LINQ de manera flexible y concisa.

## Ejemplos:

1. **Actualizar propiedades de una lista:** La siguiente expresión lambda es parte de una consulta en LINQ que trabaja sobre una lista de objetos Persona:


Expresión a analizar:

```
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);
});
```

- Se filtra la lista lstPersonas usando el método Where, que aplica una condición de filtrado a cada elemento de la lista.

- La expresión lambda "`p => p.Nombre == txtNombre.Text && p.Apellidos == txtApellidos.Text`" indica que solo se seleccionarán aquellas personas cuyo nombre y apellidos coincidan con los valores introducidos en los campos de texto txtNombre y txtApellidos. "p" representa cada objeto Persona de la lista.

.ToList():

	C# TIPS: Lambda Expression	
		Rosa María Zapata Calle Angel Quero Blanquet

- Después del filtrado, la lista resultante se convierte en una lista mediante el método `ToList()`. Esto es necesario porque el resultado de `Where` es un objeto de tipo `IEnumerable`, y al convertirlo en una lista (`List`), se puede realizar una operación como `ForEach`.

`.ForEach(p => {...}):`

- Una vez filtrada la lista, el método `ForEach` aplica a cada persona en la lista filtrada un acción, que en este caso es la actualización de las propiedades `Edad`, `Nombre` y `Apellido`.

## 2. Comprobar si algún registro cumple la condición:

Expresión a analizar:

```
if (lstPersonas.Where(p => p.Nombre == txtNombre.Text && p.Apellidos == txtApellidos.Text).ToList().Any() == false) //If not exist
```

- `lstPersonas.Where(p => p.Nombre == txtNombre.Text && p.Apellidos == txtApellidos.Text):` Filtra la lista `lstPersonas` para encontrar personas cuyo nombre y apellidos coincidan en nombre y apellidos.
- `ToList()`: Convierte el resultado filtrado en una lista.
- `.Any()`: Verifica si existe al menos un elemento en la lista filtrada. Retorna `true` si hay coincidencias y `false` si no hay.