# Escuela de Sistemas y Tecnologías

Transparencias de **ANALISTA DE SISTEMAS**

***Edición 2017 – Materia:*** **Diseño de Aplicaciones Web**

**TEMA: LINQ**

# Plantel y Contactos

- **Bedelía:**
  - Mail: bedeliasistemas@bios.edu.uy

- **Encargado de Sucursal:**
  - Pablo Castaño
  - Mail: pablocasta@bios.edu.uy

# Recursos

- **Recursos Imprescindibles:**
  - Sitio Web de material
    (comunicarse con Bedelía por usuario/contraseña).
  - Transparencias del Curso.
  - Contar con el software necesario

# Agenda

- Introducción
- LINQ to Objects
- LINQ to ADO (DataSet)

# *Introducción*

# Que es? (1)

- Language-Integrated Query
- Existente a partir de Visual Studio 2008 y de NET Framework versión 3.5
- Modelo de programación para construir consultas (***queries***) en lenguajes Microsoft.NET C# y VB

# Que es? (2)

VB.NET 9

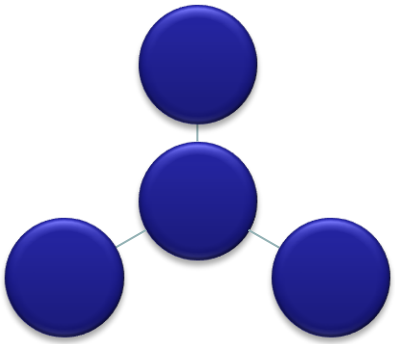
C# 3.0

.NET Language Integrated Query

LINQ a Objetos

LINQ a ADO.NET

LINQ a XML



**LINQ a ADO.NET**

LINQ a SQL

**LINQ a ADO.NET**

LINQ a Dataset

**LINQ a ADO.NET**

LINQ a Entidades

LINQ a XML

- <libro>
- <titulo/>
- <autor/>
- <precio/>
- </libro>

# Ventajas

- Hasta VS 2005 o Net Framework 3.0, las consultas con datos se expresan como cadenas sencillas, sin comprobación de tipos en tiempo de compilación, ni compatibilidad con IntelliSense.
- Además, había un lenguaje de consultas diferente para cada tipo de origen de datos:
  - bases de datos (SQL),
  - documentos XML (Xpath),
  - Etc.
- Las consultas LinQ se escriben para colecciones de objetos fuertemente tipadas, utilizando palabras clave del lenguaje y operadores con clásicos.

# Variables con Tipo Implícito

- En lugar de especificar explícitamente un tipo al declarar una variable, se puede utilizar el modificador ***var*** para indicar al compilador que deduzca y asigne el tipo de dato a dicha variable.
- Las variables declaradas como ***var*** tienen el mismo establecimiento inflexible de tipo de datos que las variables cuyo tipo se especifica explícitamente.

```
var number = 56;                    → la define como Int32
var name = “Maria Mercedes“;        → la define como String
```

# Consultas (1)

- Una consulta es una expresión que recupera datos de un origen de datos
- En una consulta ***LinQ,*** siempre se trabaja con objetos.
- Se utilizan los mismos modelos de codificación básicos para consultar y transformar datos de: documentos XML, bases de datos SQL, conjuntos de datos ADO.NET, colecciones .NET y cualquier otro formato para el que haya disponible un proveedor LinQ.
- Todas las consulta se componen de tres elementos distintos:

# Consultas (2)

- ***Obtención del origen de datos***: puede ser una variable de tipo colección, un conjunto de registros Ado, datos XML, etc.
- ***Variable de Consulta***:  se escribe una consulta con formato *Linq*, la cual se asigna a una variable
- ***Ejecución de la consulta***: mediante un comando *ForEach* se recorre la variable que se cargo en el punto anterior. Es, en este momento, cuando realmente se ejecuta la consulta *Linq* sobre el origen de datos.

# Consultas (3)

- Una ***expresión de consulta*** es una expresión como cualquier otra y se utiliza en cualquier contexto en el que una expresión de C# sea válida. Está compuesta de un conjunto de cláusulas escritas en una sintaxis declarativa. Cada cláusula contiene a su vez una o más expresiones de C#.
- Una expresión de consulta debe:
  - comenzar con una cláusula ***from***
  - finalizar con una cláusula ***select*** o ***group***.
- Entre dichas cláusulas pueden existir cláusulas opcionales: ***where, orderby, join, let,*** y otras cláusulas *from* adicionales.

# Consultas (4)

- También se puede utilizar la palabra clave ***into*** para hacer que el resultado de una cláusula ***join*** o ***group*** actúe como origen de datos para una cláusula de consulta adicional
- Una ***variable de consulta*** es la que almacena una consulta en lugar de los resultados de la misma. Es siempre un tipo ***enumerable***. Generará una secuencia de elementos cuando se recorre en iteración con la instrucción *foreach*. No almacena ninguno de los datos de resultado real que se generan en el bucle *foreach*. Cuando la instrucción *foreach* se ejecuta, los resultados de la consulta no se devuelven a través de la variable de consulta: se devuelven a través de la variable de iteración.

# Clausulas (1)

- ***from***: determina el origen de datos de la consulta
- ***group***: genera una secuencia de grupos organizada por una clave especifica. La clave puede ser de cualquier tipo de datos.
- ***into:*** se puede utilizar en una cláusula *select* o *group* para crear un identificador temporal que almacena una consulta. Esto se hace cuando se deben realizar operaciones de consulta adicionales sobre una consulta, después de una operación de agrupación o selección.

# Clausulas (2)

- ***join***: para asociar y/o combinar elementos de 2 orígenes de datos, según una comparación de igualdad entre claves especificadas en cada elemento. Se puede almacenar los resultados de dicha operación en una variable temporal mediante la palabra clave *into*.
- ***let***: para almacenar el resultado de una expresión, en una nueva variable de intervalo.
- ***orderby***: ordena los elementos (ascending – descending)

# Clausulas (3)

- ***select***: genera una secuencia del mismo tipo de objetos que los objetos contenidos en el origen de datos. Se puede utilizar para transformar datos de origen en secuencias de nuevos tipos (denominada: proyección).
- ***where***: filtra elementos en los datos de origen según una o varias expresiones.

# Operadores

| | |
|---|---|
| Operadores Lógicos | && \|\| ! |
| Operadores Relacionales | < > >= <= == |
| Resolución Ámbito | . |
| Operadores Aritmeticos | + - * / % |

# *LINQ to Objects*

# Introducción

- Espacio de nombres: ***System.Linq***
- Uso directo de consultas con cualquier colección ***IEnumerable*** o ***IEnumerable<T>*** (***List<T>***, ***Array*** etc.).
- Se debe tener definida una variable colección para utilizarla como origen de datos en la consulta.
- La colección podrá contener elementos de tipo de datos nativos de Net Framework, como objetos basados en definiciones usuarias

# IEnumerable<tipo> (1)

- Interface. Expone el enumerador, que admite una iteración simple en una colección de un tipo especificado.
- ***ToList<tipo>***() → Crea un objeto *List* <tipo> a partir de un objeto *IEnumerable* <tipo>
- ***Any***() →Determina si una secuencia contiene elementos. Devuelve un tipo boolean
- ***First***() → Devuelve el primer elemento de una secuencia.
- ***Last***() → Devuelve el último elemento de una secuencia.
- ***Cast(tipo)*** → Convierte los elementos de IEnumerable en el tipo especificado.

# IEnumerable<tipo> (2)

- ***Count***() → Devuelve el número de elementos de una secuencia.
- ***Distint*** () → Devuelve diversos elementos de una secuencia utilizando el comparador de igualdad predeterminado para comparar los valores.
- ***Max***() → Devuelve el valor máximo de una secuencia genérica. utilizando el comparador de igualdad predeterminado para comparar los valores.
- ***Min***() → ídem. Devuelve el valor mínimo.

# *LINQ to ADO (DataSet)*

# Introducción

- Espacio de nombres
  - ***System.Linq***
  - ***System.Data.Linq***
- Nos permite trabajar con colecciones de objetos DataRow (provenientes de un origen de datos de tipo ***DataTable***)
- La información puede provenir de un procedimiento almacenado (registros relacionados) o se pueden obtener desde diferentes tablas y unirlos a través de la clausula ***group*** de *Linq*.