codehouse
{ACADEMY}

codehouse
{ACADEMY}

Objetos C#

codehouse {ACADEMY}

1. String

- ✓ El valor es un texto
- ✓ Internamente es una colección de objetos del tipo Char
- ✓ Length para obtener la longitud de objetos Char

1.1. string VS System.String

- ✓ String y string son equivalentes
- ✓ La clase String ofrece múltiples métodos para crear, manipular y comparar cadenas

Declaración e inicialización de cadenas
Inmutabilidad de los objetos de cadena
Literales de cadena regulares y textuales
Secuencias de escape de cadena
Cadenas de formato
Subcadenas
Acceso a caracteres individuales
Cadenas nulas y cadenas vacías
Cadenas, métodos de extensión y LINQ

codehouse
{ACADEMY}

2. StringBuilder

- ✓ Hay que tener en cuenta las operaciones con cadenas para el performance
- ✓ La clase StringBuilder crea un bufer de cadena que proporciona mejor rendimiento para las manipulaciones
- ✓ Permite reasignar caracteres individuales

codehouse
{ACADEMY}

3. Listas

3.1. ArrayList

- ✓ Permite agregar elementos dinámicamente en ejecución. Implementa la interfaz IList.
- ✓ En la actualidad no se recomienda usarla. Se recomienda List<T>
- ✓ No ofrece el mejor rendimiento. En su lugar:
  - ➢ Colección heterogénea de objetos List<Object>
  - ➢ Colección homogénea de objetos List<T>

codehouse {ACADEMY}

4. Linq

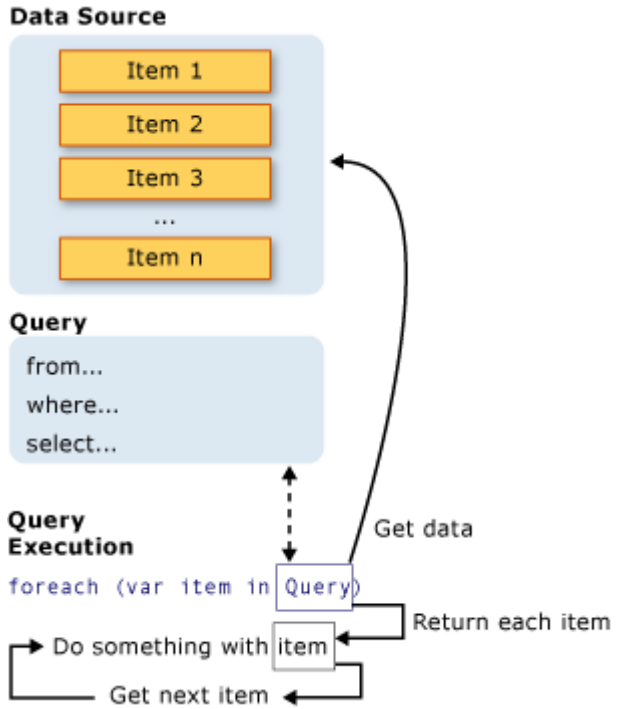
4.1. Introducción

- ✓ LINQ o Language Integrated Query
- ✓ Conjunto de herramientas de Microsoft para realizar todo tipo de consultas a distintas fuentes de datos: xml, bases de datos, etc
- ✓ Hace uso de un tipo de funciones propias
- ✓ Se consigue el uso de un mismo lenguaje para todo tipo de tareas con datos

codehouse
{ACADEMY}

4.2. Tres partes de una consulta

- ✓ Obtener el origen de datos
- ✓ Crear la consulta
- ✓ Ejecutar la consulta



codehouse {ACADEMY}

4.3. Lambda

- ✓ Sintaxis más directa
- ✓ Expresiones de métodos apoyados en operadores lambda
- ✓ Se puede llamar directamente a funciones where, join, select, etc. desde el objeto

Query Syntax

```
var vendorQuery = from v in vendors
                  where v.CompanyName.Contains("Toy")
                  orderby v.CompanyName
                  select v;
```

Method Syntax

```
var vendorQuery = vendors
              .Where(v => v.CompanyName.Contains("Toy"))
              .OrderBy(v=> v.CompanyName);
```

codehouse
{ACADEMY}

4.4. Ejecución retardada

- ✓ Se declara una variable anónima y no se fuerza la carga mediante ToList(), Select(), First(), etc.
- ✓ La consulta queda declarada, pero no lanzada para su posterior uso
- ✓ CUIDADO con el manejo de la ejecución retardada

codehouse
{ACADEMY}

4.5. Claúsulas

- ✓ Iniciar una consulta con la cláusula
  - ➢ From

- ✓ Finalizar una expresión de consulta
  - ➢ Select: Crea una proyección a partir de una secuencia
  - ➢ Group: Agrupación

- ✓ Filtrar, ordenar y combinar
  - ➢ Join: Enlazar
  - ➢ Into
  - ➢ Let
  - ➢ Where: Restricción Filtra los elementos de una secuencia
  - ➢ Orderby: Ordenación, (Reverse)

codehouse
{ACADEMY}

Otros operadores:

- ✓ Matemáticos
  - ➢ Count
  - ➢ Sum
  - ➢ Max
  - ➢ Average

- ✓ Elemento
  - ➢ ElementAt. Devuelve el elemento de un índice determinado en una secuencia
  - ➢ ElementAtOrDefault. Devuelve el elemento de un índice determinado en una secuencia o un valor predeterminado si el índice está fuera del intervalo
  - ➢ First. Devuelve el primer elemento de una secuencia
  - ➢ FirstOrDefault. Devuelve el primer elemento de una secuencia o un valor predeterminado si no se encuentra ningún elemento
  - ➢ Last. Devuelve el último elemento de una secuencia
  - ➢ LastOrDefault. Devuelve el último elemento de una secuencia o un valor predeterminado si no se encuentra ningún elemento
  - ➢ Single. Devuelve el único elemento de una secuencia
  - ➢ SingleOrDefault. Devuelve el único elemento de una secuencia o un valor predeterminado si no se encuentra ningún elemento

codehouse
{ACADEMY}

- ✓ Generación
  - ➢ Empty. Genera una secuencia vacía
  - ➢ Range. Genera una secuencia dado un rango
  - ➢ Repeat. Genera una secuencia repitiendo un elemento un número determinado de veces

- ✓ Particiones de datos
  - ➢ Skip. Devuelve una secuencia que omite un número determinado de elementos
  - ➢ SkipWhile. Devuelve una secuencia que omite elementos que no cumplen una expresión
  - ➢ Take. Devuelve una secuencia que toma un número determinado de elementos
  - ➢ TakeWhile. Devuelve una secuencia que toma elementos que cumplen una expresión

- ✓ Cuantificadores
  - ➢ All. Determina si todos los elementos de una secuencia cumplen una condición
  - ➢ Any. Determina si los elementos de una secuencia cumplen una condición
  - ➢ Contains. Determina si una secuencia contiene un elemento determinado

codehouse
{ACADEMY}

- ✓ Conversión
  - Cast. Convierte elementos de una secuencia en otro tipo dado
  - OfType. Filtra los elementos de una secuencia de un determinado tipo
  - ToArray. Devuelve un Array de una secuencia
  - ToDictionary. Devuelve un Dictionary de una secuencia
  - ToList. Devuelve un List de una secuencia
  - ToLookup. Devuelve un Lookup de una secuencia
  - ToSequence. Devuelve un IEnumerable de una secuencia

codehouse {ACADEMY}

5. Excepciones

5.1. Introducción

- ✓ Evento que interrumpe el flujo normal de operación
- ✓ Clase System.Exception:
  - ✓ InnerException
  - ✓ Message
- ✓ Tipos de excepciones:
  - ➢ Implícitas
  - ➢ Explícitas

codehouse {ACADEMY}

5.2. Captura de excepciones

- ✓ Try-Catch-Finally
- ✓ Sentencia throw
- ✓ Aspectos a evitar:
  - ➢ No se deben usar para cambiar el flujo
  - ➢ No se deben devolver como parámetro o valor devuelto
  - ➢ No generar de forma intencionada
  - ➢ No crear excepciones que solo se puedan reproducir en depuración

codehouse {ACADEMY}

5.3. Custom Exceptions

- ✓ Las nuevas clases deben ser serializables
- ✓ Deben cumplir:
  - ✓ Ser simples de usar y entender
  - ✓ Debe separar el código del tratamiento de excepciones del código normal
  - ✓ Implementar un tratamiento uniforme de las excepciones

codehouse
{ACADEMY}