Diseño Basado en Componentes

Colecciones en VB.NET

Ingeniería Informática
Universidad Carlos III de Madrid

Colecciones en VB.NET

- Especialización de los **arrays** (optimizado o especializado para ciertas tareas).
- Objeto que internamente gestiona un array de una manera un tanto especial.
- Clases significativas (*colecciones*) del espacio de nombres **System.Collections** (no es necesario importarlo, pues se incluye por defecto en el IDE):
  - **ArrayList**: proporciona una colección cuyo array es redimensionado dinámicamente.
  - **Hashtable**: proporciona una colección cuyo array contiene elementos que se basan en una combinación de clave-valor, facilitando el acceso al realizarse mediante la clave.

Colecciones en VB.NET
Interfaces

- Las clases integrantes de **System.Collections** implementan en mayor o menor grado, un conjunto común de interfaces:
  - IEnumerable: proporciona el soporte para recorrer colecciones de valores.
  - ICollection: permite manipular el tamaño, gestionar enumeradores, etc., de listas de valores.
  - IList: referencia a una lista de valores que puede ordenarse.
  - ICloneable: permite la creación de copias exactas e independientes de objetos.
- Para crear nuestro propio tipo de colección habría que heredar una clase colección existente y/o la implementación de alguna de las interfaces de **System.Collections**.

Colecciones en VB.NET
ArrayList (I)

- Instanciación de objetos **ArrayList**:

```
' Crear una lista sin elementos
Dim alEstaciones As New ArrayList()
' Crear una lista indicando el número de elementos,
' pero sin darles valor
Dim alDatos As New ArrayList(3)
' Crear una lista utilizando una colección dinámica
Dim alLetras As New ArrayList(New String() {"a", "b", "c"})
```

- Capacidad y valores en una colección ArrayList:
  - Propiedad **Capacity**: número de elementos del array subyacente que contiene este objeto.
  - Propiedad **Count**: número de elementos del array a los que se ha asignado valor a través de los métodos **Add(Value)** o **AddRange(Collection)**.

Colecciones en VB.NET
ArrayList (II)

- Agregar valores a un ArrayList:
  - **Add(Value)**: añade el valor representado por **Value**.
  - **Insert(Position, Value)**: inserta el valor **Value** en la posición Position del array, desplazando el resto de valores una posición adelante.
- Borrado de elementos en un ArrayList:
  - **Remove(Value)**: elimina el elemento del array que corresponde a **Value**.
  - **RemoveAt(Position)**: elimina el elemento del array situado en el índice **Position**.
  - **Clear()**: elimina todos los elementos del objeto ArrayList.

Colecciones en VB.NET
ArrayList (III)

- Formas de recorrer y obtener valores de un ArrayList:
  - Bucle **For…Next** y la propiedad **Count** del objeto ArrayList.

```
Dim alLetras As ArrayList()
…
Dim iContador As Integer
For iContador = 0 To (alLetras.Count - 1)
   Console.WriteLine("Elemento actual {0}, valor: {1}", _
   iContador, alLetras(iContador))
Next
```

  - Recorrer el array con un **enumerador**: objeto de la interfaz **IEnumerator**, proporcionado por el método **GetEnumerator()**.

```
Dim alLetras As ArrayList()
…
Dim enumerador As IEnumerator
enumerador = alLetras.GetEnumerator()
While enumerador.MoveNext()
   Console.WriteLine("Elemento de la lista: {0}", enumerador.Current())
End While
```


Colecciones en VB.NET
Hashtable (I)

- El acceso a los valores del array se realiza a través de una clave asociada a cada elemento.
- Un objeto Hashtable ofrece la ventaja de no tener que saber la posición en la que se encuentra cada valor: cada posición tiene asignado un nombre clave.

```
' Declarar colección Hashtable
Dim htCliente As Hashtable
htCliente = New Hashtable()
' Añadir valores
htCliente.Add("ID", 2245321)
htCliente.Add("Nombre", "Pedro")
htCliente.Add("Apellidos", "Santana")
htCliente.Add("Domicilio", "C/ Espinar, 42")
htCliente.Add("Edad", 33)
htCliente.Add("Credito", 600)
```


Colecciones en VB.NET
Hashtable (II)

```
Console.WriteLine("El objeto Hashtable tiene {0} elementos", _
                  htCliente.Count)
Console.WriteLine()
' Obtener algunos valores
Console.WriteLine("Obtener algunos valores del objeto Hashtable")
Console.WriteLine("Domicilio: {0} ", htCliente.Item("Domicilio"))
Console.WriteLine("Nombre: {0} ", htCliente("Nombre"))
Console.WriteLine("Credito: {0} ", htCliente("Credito"))
Console.WriteLine()
' Recorrer el array al completo
Console.WriteLine("Recorrer objeto Hashtable con un enumerador")
Dim enumerador As IDictionaryEnumerator
enumerador = htCliente.GetEnumerator()
While enumerador.MoveNext()
    Console.WriteLine("Clave: {0} / Valor: {1}", _
                      enumerador.Key, enumerador.Value)
End While
Console.ReadLine()
```

Colecciones en VB.NET
Hashtable (III)

- Operaciones con colecciones Hashtable:
  - **Add(Key, Value)**: añade el elemento a la colección con la clave **Key** y el valor **Value**.
  - **Remove(Key)**: elimina el elemento del objeto cuya clave coincide con **Key**.
  - **Clear()**: elimina el contenido completo de la colección.
  - **ContainsKey(Key)**: comprueba si la clave **Key** se encuentra en la colección.
  - **ContainsValue(Value)**: comprueba si el valor **Value** se encuentra en la colección.

Colecciones en VB.NET
Hashtable (IV)

- Las propiedades **Keys** y **Values** de la clase Hashtable devuelven un objeto de la interfaz ICollection.

```
' Declarar colección Hashtable
Dim htCliente As Hashtable
htCliente = New Hashtable()
' Añadir valores
htCliente.Add("ID", 2245321)
htCliente.Add("Nombre", "Pedro")
htCliente.Add("Apellidos", "Santana")
htCliente.Add("Domicilio", "C/ Espinar, 42")
htCliente.Add("Edad", 33)
htCliente.Add("Credito", 600)
```


Colecciones en VB.NET
Hashtable (V)

```
' Obtener un array con las claves, y recorrer con un enumerador
Dim aClaves As ICollection
Dim enumerador As IEnumerator
aClaves = htCliente.Keys
enumerador = aClaves.GetEnumerator()
Console.WriteLine("Claves del objeto Hashtable")
RecorrerEnumerador(enumerador)
' Obtener un array con los valores, y recorrer con un enumerador
Dim aValores As ICollection
Dim enumVal As Ienumerator
aValores = htCliente.Values
enumVal = aValores.GetEnumerator()
Console.WriteLine("Valores del objeto Hashtable")
RecorrerEnumerador(enumVal)
Console.ReadLine()
```


Colecciones en VB.NET
Hashtable (VI)

```
Public Sub RecorrerEnumerador(ByVal enumerador As IEnumerator)
    While enumerador.MoveNext
        Console.WriteLine(enumerador.Current)
    End While
    Console.WriteLine()
End Sub
```


Diseño Basado en Componentes
Colecciones en VB.NET
Information Engineering
Ingeniería Informática
Universidad Carlos III de Madrid
Diseño Basado en Componentes
Curso 2008 / 09