Системные коллекции

№ урока: 2 **Курс:** C# Professional

Средства обучения: Компьютер с установленной Visual Studio

Обзор, цель и назначение урока

Надлежащая организация и хранение данных является первоочередной и наиболее важной задачей для каждого программиста. Для решения этой проблемы .NET Framework предлагает широкий спектр наборов, позволяющих хранить и эффективно обрабатывать данные. На данном уроке демонстрируется все разнообразие основных классов для группировки связанных объектов и последующей их обработки.

Изучив материал данного занятия, учащийся сможет:

- Управлять группами однотипных и разнотипных данных в .NET приложениях с использованием наборов и специализированных наборов.
- Повышать производительность и эффективность контроля типов в .NET приложениях с использованием обобщенных наборов.
- В зависимости от поставленной задачи и ее специфики, своевременно и правильно использовать специализированные наборы для оптимизации производительности.

Содержание урока

1. Использование `ArrayList` и сбор элементов данных. Предназначение и использование интерфейса `IComparer`.
2. Работа с последовательными списками (`Queue`, `Stack`).
3. Работа со словарями (`Hashtable`, `SortedList`, `ListDictionary`, `HybridDictionary`, `OrderedDictionary`).
4. Роль интерфейса `IEqualityComparer`.
5. Применение специализированных наборов.
6. Работа с двоичными значениями с помощью `BitArray` и `BitVector32`.
7. Использование методов `CreateMask` и `CreateSection`.
8. Хранение строк в наборах (`StringCollection`, `StringDictionary`, `NameValueCollection`).
9. Обобщенные наборы.

Резюме

- Наборы – классы, предназначенные для группировки связанных объектов, управления ими и обработки их в циклах, – являются одним из основных инструментов программиста.
- В .NET Framework в пространстве имен `System.Collections` реализовано большое количество специализированных наборов, которые значительно упрощают и ускоряют процесс разработки.
- `ArrayList` – простой, поддерживающий индексирование и изменение размера набор объектов. Является простым неупорядоченным контейнером для объектов любого типа. В наборах разрешается хранить значимые типы, но сначала их необходимо преобразовать к ссылочному типу, выполнив операцию упаковки (boxing). Метод `AddRange` позволяет добавлять диапазоны (или группу элементов) любых объектов, поддерживающих интерфейс `ICollection` (включая массивы, объекты `ArrayList` и большинство других наборов). Методы `Add` и `AddRange` добавляют элементы в конец набора. Поскольку наборы `ArrayList` являются динамическими, они поддерживают вставку объектов в заданном положении. Для этого `ArrayList` предоставляет методы `Insert` и `InsertRange`. Интерфейс `IEnumerable` унифицирует перебор элементов набора в цикле. .NET Framework также поддерживает общий интерфейс – образец API наборов. Этот интерфейс называется `ICollection` и происходит от интерфейса `IEnumerable`. Это означает, что любой набор, поддерживающий интерфейс `ICollection`, также поддерживает интерфейс `IEnumerable`. Интерфейс `ICollection` содержит в себе 3 свойства(`Count`, `IsSynchronized`, `SyncRoot`) и один метод (`CopyTo`).

CyberBionic systematics

CyberBionic Systematics ® 2015
19 Mariny Raskovoy Str., 5 floor
Kyiv, Ukraine

t. +380 (44) 361-8473
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com

- .NET Framework поддерживает еще один интерфейс для доступа к элементам списка: `IList`. Этот интерфейс наследуется непосредственно от `ICollection`. Если класс поддерживает интерфейс `IList`, он также поддерживает интерфейсы `ICollection` и `IEnumerable`.
- Последовательные списки рекомендуется использовать в том случае, когда необходимо обрабатывать списки объектов последовательно, а не в произвольном порядке. Последовательные списки представлены двумя классами (`Queue` и `Stack`).
- `Queue` – набор объектов, организованный по принципу «первым вошел, первым вышел» (FIFO; first-in, first-out). В отличии от `ArrayList`, в котором обращение к элементу и удаление его из набора представлены различными операциями, `Queue` объединяет эти операции в комбинированном методе `Dequeue`. Эти операции исходно логически связаны в силу специфики класса `Queue`. Создав экземпляр класса, можно вызывать метод `Enqueue` для добавления элементов в очередь и метод `Dequeue` для удаления элементов из нее. Класс `Queue` позволяет добавлять в список дублирующиеся элементы и `null`-значения, поэтому стандартные методы `Dequeue` и `Peek` не помогут определить, пуста ли очередь `Queue`.
- `Stack` – набор объектов, организованный по принципу «последним вошел, первым вышел» (LIFO; last-in, first-out). Работу данного класса можно сравнить с колодой карт из которой карты можно убрать и ложить обратно только сверху. Работа с классом `Stack` похожа на работу с классом `Queue`: создают экземпляр класса `Stack`, затем для добавления элементов в стек вызывают метод `Push`, а для удаления – метод `Pop`. Оба класса `Stack` и `Queue` поддерживают метод `Peek` для просмотра следующего элемента в наборе без его удаления.
- `Hashtable` – набор пар объектов «имя-значение», предоставляющих доступ к элементам как по имени, так и по индексу в наборе. Чтобы получить доступ к данным, уже добавленным в словарь, достаточно вызвать индексатор с требуемым ключом. Если из словаря требуется вывести значения, следует указать итератору, что он работает с объектами типа `DictionaryEntry`. Класс `Hashtable` требует уникальности хэш-кодов, а не связанных с ними значений. Если же попытаться сохранить разные значения с одним и тем же ключом, то второе значение заменит первое. Класс `Hashtable` чувствителен к регистру. Существенным недостатком данного класса является высокое потребление ресурсов, при использовании маленьких наборов (меньше десяти элементов) это снижает производительность.
- Интерфейс `IEqualityComparer` предоставляет возможность реализовать настраиваемое сравнение коллекций в отношении равенства. Это означает, что можно создать свое собственное определение равенства и указать, что это определение должно использоваться для типа коллекции, которая принимает интерфейс `IEqualityComparer`. В .NET Framework конструкторы типов коллекций `Hashtable`, `NameValueCollection` и `OrderedDictionary` принимают этот интерфейс. Данный интерфейс поддерживает только сравнения в отношении равенства. Настройка сравнения для сортировки или упорядочения реализуется с помощью интерфейса `IComparer`.
- `SortedList` – упорядоченный набор пар объектов «имя-значение». Чтобы упорядочить коллекцию, представленную `SortedList` вызов метода `Sort` не требуется. `SortedList` сам упорядочит элементы в момент добавления
- `ListDictionary` – набор, подходящий для хранения небольших списков объектов. Он оптимален для небольших наборов (до 10-ти элементов), поскольку устроен как простой массив.
- `HybridDictionary` – набор, в котором элементы хранятся в `ListDictionary`, если их мало, либо в `Hashtable`, если их много. Используется в тех случаях, когда определить размер коллекции изначально невозможно.
- `BitArray` – компактный набор значений типа `Boolean`. Класс `BitArray` поддерживает нединамическое изменение размера. Размер набора необходимо указывать при создании экземпляра класса `BitArray`. После этого для изменения размера можно воспользоваться свойством `Length`. Элементы набора `BitArray` принимают одно из двух значений: `true` или `false`, поэтому сама концепция добавления или удаления элементов к такому набору неприменима.
- Структура `BitVector32` очень удобна для управления отдельными битами больших чисел. `BitVector32` хранит свои данные как 32-разрядное целое число. Все операции над объектами `BitVector32` в действительности изменяют значение целого внутри структуры. Структура `BitVector32` позволяет последовательно создавать битовые маски, вызывая статический метод `CreateMask`. Эти маски можно использовать вместе с индексатором

CyberBionic systematics

CyberBionic Systematics ® 2015
19 Mariny Raskovoy Str., 5 floor
Kyiv, Ukraine

t. +380 (44) 361-8473
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com

структуры BitVector32 для установки или получения значения отдельного бита, соответствующего маске. Также коллекция BitVector32 очень удобна для работы с отдельными битами и поддерживает упаковку битов. Упаковка часто позволяет сэкономить память, необходимую для хранения маленьких чисел.

- StringCollection – простой, поддерживающий изменение размера, набор строк. Код, добавляющий строки в набор, выглядит так же, как и в приведенном выше примере с использованием ArrayList. Единственное отличие в том, что попытка добавления объекта с типом, отличным от строкового, влечет ошибку компиляции. К тому же теперь, получая строки из набора, вы работаете не с объектами, а именно со строками, что устраняет необходимость приведения типов.
- StringDictionary – набор пар строк «имя-значение», предоставляющих доступ к элементам как по имени, так и по индексу в наборе. Он работает как Hashtable, но и ключ, и значение должны быть строками. Важно понимать, что по умолчанию ключи для объектов StringDictionary нечувствительны к регистру, поэтому ключи "Fourth" и "FOURTH" эквиваленты.
- NameValueCollection – набор пар строк «имя-значение», предоставляющих доступ к элементам как по имени, так и по индексу в наборе. В классе NameValueCollection допускается несколько значений, соответствующих одному ключу, к тому же, значения можно получать не только по ключу, но и по индексу. Таким образом, при работе с классом NameValueCollection можно хранить несколько значений с одним ключом. Метод Add позволяет делать это.
- Класс CollectionUtil поддерживает создание объектов Hashtable и SortedList, нечувствительных к регистру. Для создания этих объектов достаточно вызвать метод CreateCaseInsensitiveHashtable или CreateCaseInsensitiveSortedList.
- Обобщения – это типы, которые принимают имена других типов в качестве параметров, и используют их. Вместо того, чтобы создавать строго типизированные наборы для каждого типа, достаточно создать набор, которые можно настроить на работу с любым типом.
- Класс-обобщение LinkedList имеет одну особенность – реализацию перечислителя ILinkedListEnumerator, что позволяет перечислять значения списка без использования объектов LinkedListNode.

Закрепление материала

- Что представляет собой коллекция ArrayList?
- В каких случаях лучше использовать Hashtable, ListDictionary, HybridDictionary, OrderedDictionary?
- Благодаря какому интерфейсу в коллекциях можно реализовать настраиваемое сравнение в отношении равенства? Для каких коллекций данный интерфейс можно применить?
- Как получить все значения, соответствующие отдельному ключу в наборе NameValueCollection?
- Перечислите основные отличия между NameValueCollection и StringDictionary?
- Сраните ArrayList и StringCollection опишите основные преимущества StringCollection.
- Перечислите основные коллекции, которые не чувствительны к регистру. Как можно исправить данную проблему?
- В чем основное преимущество обобщенных коллекций? Какие обобщенные коллекции вам известны?

Дополнительное задание

Используя класс SortedList, создайте небольшую коллекцию и выведите на экран значения пар «ключ- значение» сначала в алфавитном порядке, а затем в обратном.

Самостоятельная деятельность учащегося

Задание 1
Выучите основные конструкции и понятия, рассмотренные на уроке.

CyberBionic systematics
CyberBionic Systematics ® 2015
19 Mariny Raskovoy Str., 5 floor
Kyiv, Ukraine
t. +380 (44) 361-8473
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com

Задание 2
Создайте коллекцию, в которую можно добавлять покупателей и категорию приобретенной ими продукции. Из коллекции можно получать категории товаров, которые купил покупатель или по категории определить покупателей.

Задание 3
Несколькими способами создайте коллекцию, в которой можно хранить только целочисленные и вещественные значения, по типу «счет предприятия – доступная сумма» соответственно.

Задание 4
Создайте коллекцию типа `OrderedDictionary` и реализуйте в ней возможность сравнения значений ключей.

Задание 5
Зайдите на сайт MSDN.
Используя поисковые механизмы MSDN, найдите самостоятельно описание темы по каждому примеру, который был рассмотрен на уроке, так, как это представлено ниже, в разделе «Рекомендуемые ресурсы», описания данного урока. Сохраните ссылки и дайте им короткое описание.

Рекомендуемые ресурсы

MSDN: Пространство имен `System.Collections`
http://msdn.microsoft.com/ru-ru/library/k166wx47.aspx

MSDN: Введение в универсальные шаблоны. (Руководство по программированию на C#)
http://msdn.microsoft.com/ru-ru/library/0x6a29h6(v=VS.100)

MSDN: Коллекции и структуры данных
http://msdn.microsoft.com/ru-ru/library/7y3x785f.aspx

MSDN: Когда следует использовать универсальные коллекции
http://msdn.microsoft.com/ru-ru/library/ms172181.aspx

MSDN: Пространство имен `System.Collections.Generic`
http://msdn.microsoft.com/ru-ru/library/system.collections.generic.aspx

CyberBionic systematics
CyberBionic Systematics ® 2015
19 Mariny Raskovoy Str., 5 floor
Kyiv, Ukraine
t. +380 (44) 361-8473
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com