C# Professional

Пользовательские коллекции

После урока обязательно

Повторите этот урок в видео формате на ITVDN.com

Проверьте как Вы усвоили данный материал на TestProvider.com

Пользовательские коллекции

Collection

Коллекция – это класс, предназначенный для группировки связанных объектов, управления ими и обработки их в циклах.

Коллекции являются важным инструментом программиста, но решение о их применении не всегда оказывается очевидным.

Применение коллекций

Коллекции стоит применять, если:

- Отдельные элементы используются для одинаковых целей и одинаково важны.
- На момент компиляции число элементов не известно или не зафиксировано.
- Необходима поддержка операции перебора всех элементов.
- Необходима поддержка упорядочивания элементов.
- Необходимо использовать элементы из библиотеки, от которой потребитель ожидает наличия типа коллекции.

IEnumerable

Методы интерфейса `IEnumerable`:

`IEnumerator` GetEnumerator()- возвращает перечислитель, который можно использовать для навигации по коллекции.

IEnumerator

Свойства интерфейса IEnumerator:

object Current { get; } – возвращает текущий элемент коллекции.

Методы интерфейса IEnumerator:

bool MoveNext() – перемещает перечислитель на следующий элемент коллекции.

void Reset() – возвращает перечислитель на начало коллекции.

IEnumerable<T>

`IEnumerable<T>` – унаследован от `IEnumerable`

Методы интерфейса `IEnumerable<T>`:

`IEnumerator<T> GetEnumerator()` – возвращает обобщенный перечислитель, который можно использовать для навигации по коллекции.

IEnumerator<T>

`IEnumerator<T>` унаследован от `IDisposable` и `IEnumerator`

Свойства интерфейса `IEnumerator<T>`:

`T Current { get;}` – возвращает текущий элемент коллекции.

ICollection

`ICollection` унаследован от `IEnumerable`.

Свойства интерфейса `ICollection`:

- `int Count { get; }` – возвращает количество элементов, хранящихся в коллекции.
- `bool IsSynchronized { get; }` – признак синхронизации доступа к коллекции.
- `object SyncRoot { get; }` – объект синхронизации доступа к коллекции.

Методы интерфейса `ICollection`:

- `void CopyTo(Array array, int index)` – выполняет копирование элементов коллекции в массив, начиная с указанного индекса.

ICollection<T>

`ICollection<T>` унаследован от `IEnumerable<T>` и `IEnumerable`

Свойства интерфейса `ICollection<T>`:

- `int Count { get; }` – возвращает количество элементов, хранящихся в коллекции.
- `bool IsReadOnly { get; }` – показывает, является ли коллекция доступной только для чтения.

Методы интерфейса `ICollection<T>`:

- `void Add(T item)` – позволяет помещать элементы в коллекцию.
- `void Clear()` – очищает содержимое коллекции.
- `bool Contains(T item)` – определяет, содержится ли в коллекции указанное значение.
- `void CopyTo(T[] array, int arrayIndex)` – выполняет копирование элементов коллекции в массив, начиная с указанного индекса.
- `bool Remove(T item)` – удаляет из коллекции первый элемент, значение которого соответствует значению аргумента.

IList

`IList` Унаследован от `ICollection` и `IEnumerable`

Свойства интерфейса `IList`:

- `bool IsFixedSize { get; }` – показывает, является ли размер списка фиксированным.
- `bool IsReadOnly { get; }` - показывает, является ли список доступным только для чтения.
- `object this[int index] { get; set; }` – индексатор, позволяющий помещать либо извлекать значения по указанному индексу.

Методы интерфейса `IList`:

- `int Add(object value)` – позволяет помещать элементы в список.
- `void Clear()` – очищает содержание списка.
- `bool Contains(object value)` – определяет, содержится ли в списке указанное значение.
- `int IndexOf(object value)` – возвращает индекс элемента с указанным значением.
- `void Insert(int index, object value)` – помещает элемент в список по указанному индексу.
- `void Remove(object value)` – удаляет из списка первый элемент, значение которого соответствует значению аргумента.
- `void RemoveAt(int index)` – удаляет элемент по указанному индексу.

yield

- Блок, в котором содержится ключевое слово **`yield`**, расценивается компилятором, как блок итератора.

- Ключевое слово **`return`** используется для предоставления значения объекту перечислителя.

- Ключевое слово **`break`** используется для обозначения конца итерации.

foreach

Циклическая конструкция `foreach` позволяет выполнять навигацию по коллекции, используя реализации интерфейсов `IEnumerable` и `IEnumerator`.

Q&A

Информационный видеосервис для разработчиков программного обеспечения

ITVDN

IT VIDEO DEVELOPERS NETWORK

TestProvider PROMETRIC TEST CENTER CyberBionic systematics Microsoft Partner Silver Learning Windows Azure Cloud Partner EBA European Business Association