

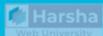
Section 24: Tuples

28 July 2024 18:21

Tuples

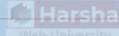
Harsha Vardhan

- › UI Expert
- › .NET Expert
- › Module Lead
- › Corporate Instructor

 Harsha
Web University

Agenda

- 1 **Tuple class**
What is Tuple class and how to use it. New Feature of C# 4.0.
- 2 **Value Tuples**
Creating tuples easily. New feature of C# 7.1.
- 3 **Deconstruction**
Copying elements from value tuple into local variables.
- 4 **Discards**
Skipping elements while deconstruction.

 Harsha
Web University

The Tuple is an object that can contain multiple values at the same time so it is an alternative to anonymous object but the benefit is that you can use the tuples as a parameter type or return type.

What

- › The System.Tuple class represents a set of values of any data type.
- › Introduced in C# 4.0.
- › Useful to return multiple values from a method (or) to pass multiple values to a method.
- › Represents a set of values quickly without creating a separate class.
- › Alternative to anonymous objects (to be used as parameter types / return types).

How

Step 1: Object of Tuple class

```
var referenceVariable = new Tuple<type1, type2, ...>() { value1, value2, ... };
```

Step 2: Accessing Elements

```
referenceVariable.Item1 //returns value1  
referenceVariable.Item2 //returns value2  
...
```

Tuple

```
Item1 = value1  
Item2 = value2
```

From a method you want to return multiple values in that cases one option is that you can

create a separate class with the specified properties and specified that class as the written

type of the method for example there is a method called get person details and from this method you want to return

person name and age so the option number one is that create a separate class called person details with two properties that is person name and age and for this get person details method specify the written type as person details and inside the method you create and return an object of person details class this is one option to return multiple values but it's a multi-step process and a bit lengthy process right.

step one you should create a separate class called person details and specific that class as the written

type of the particular method and you should make a separate object for the person details class

and return that object from the get person details method so it's a lengthy process but quickly you want to return an object

that contains multiple values for example it should include person name and age here.

second option is that we can return an anonymous object and specify the return type as object that means system.object class but the problem is after receiving the value in the calling portion it will be difficult to identify what

properties are exist in the object because the written type is system.object that is why it is not recommended to written anonymous objects from any method you can use anonymous objects within the same method but not across the methods that means should not be used as method parameter or written type of the particular method.

the third option is that you can use out keyword.

Fourth option, use tuple class

The screenshot shows a presentation slide titled "'Tuple' class" with a navigation bar at the top. The slide is divided into two main sections: "What" and "How".

What:

- > The System.Tuple class represents a set of values of any data type.
- > Introduced in C# 4.0.
- > Useful to return multiple values from a method (or) to pass multiple values to a method.
- > Represents a set of values quickly without creating a separate class.
- > Alternative to anonymous objects (to be used as parameter types / return types).

How:

Step 1: Object of Tuple class

```
var referenceVariable = new Tuple<type1, type2, ...>() { value1, value2, ... };
```

Step 2: Accessing Elements

```
referenceVariable.Item1 //returns value1  
referenceVariable.Item2 //returns value2  
...
```

Tuple

Item1 = value1
Item2 = value2

4:02



- › Tuple stores only a set of values (of any data type); but doesn't store property names.
 - › So you should access them as Item1, Item2 etc., which doesn't make sense some times.
- › Tuple supports up to 8 elements only by default.
 - › You can store more than 8 values by using nested tuples (tuple inside tuple).
- › Tuples are mainly used to pass multiple values to a method as parameter; and also return multiple values from a method.

Value Tuples

What

- › 'Value Tuples' are advancement to 'Tuple' class with simplified syntax.
- › Introduced in C# 7.1.

How**Step I: Creating Value Tuple**

```
(type fieldName1, type fieldName2, ...) referenceVariable = (value1, value2, ...);
```

```
field1 = value1  
field2 = value2
```

So what is the big complaint about the Tuple class? The process of creating the Tuple class's object and initializing the values into that object is a big process. That means it's a bit lengthy syntax.

And also each value is represented as item1, item2, item3 etc., properties; instead of representing actual field names such as customerName or phoneNumber. These two

problems are overcomed in C# 7.1 by introducing value tuples.

The slide has a blue header bar with the title "Value Tuples" and a button "Next: Deconstruction".

What

- > 'Value Tuples' are advancement to 'Tuple' class with simplified syntax.
- > Introduced in C# 7.1.
- > Supports unlimited values.
- > You will access elements with real field names; instead of Item1, Item2 etc.
- > Can be used as method parameters / return value; much like Tuple class.

How

Step 1: Creating Value Tuple
`(type fieldName1, type fieldName2, ...) referenceVariable = (value1, value2, ...);`

Step 2: Accessing Elements
`referenceVariable.fieldName1 //returns value1`
`referenceVariable.fieldName2 //returns value2`
...

Harsha
Web University
Value Tuple

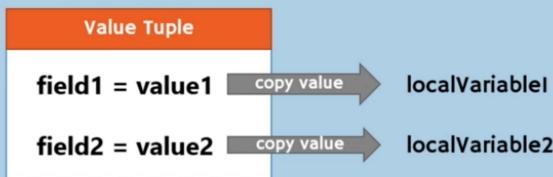
field1 = value1
field2 = value2

Value tuple is the best for returning multiple values at a time from a method.

Desconstruction

What

- Deconstruction allows you to assign elements of value tuple into individual local variables.

**How**

`(type fieldName1, type fieldName2, ...) referenceVariable = (value1, value2, ...);`

Step 1: Create Value Tuple

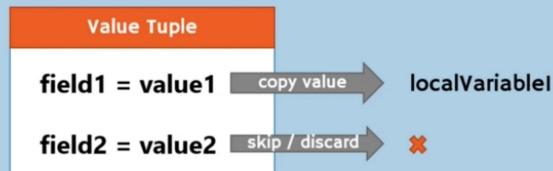
`(type variableName1, type variableName2, ...) = referenceVariable;`

Step 2: Deconstruction

Discards

What

- Discard allows you to skip a value which you don't require, by using underscore (_).

**How**

`(type fieldName1, type fieldName2) referenceVariable = (value1, value2);`

Step 1: Create Value Tuple

`(type variableName1, _) = referenceVariable;`

Step 2: Deconstruction with Discard