MBARK T3STO
259 Followers

Aug 9, 2021 · 3 min read

The hat (^) and range (..) operators in C#

The hat operator (^) and range operator (. .) provide a different syntax for accessing elements in an array: Span, or ReadOnlySpan. The range operator is used to specify the start and end of a range for a sequence.




```
static void Main(string[] args)
{
  var people = new string[] { "Jane", "Jean", "Grey", "Marcus",
"Theophilus", "Keje" };
  var firstFour = GetFirstFourPersons(people);

  foreach (var person in firstFour)
  {
    Console.WriteLine(person);
  }
}

static string[] GetFirstFourPersons(string[] people)
{
  var result = new string[4];
  for (int i = 0; i < 4; i++)
  {
    result[i] = people[i];
  }
  return result;
}

// result:
// Jane
// Jean
// Grey
// Marcus
```

We can rewrite it using the range operator, passing the range operand inside `[` and `]`.

```
static void Main(string[] args)
{
  var people = new string[] { "Jane", "Jean", "Grey", "Marcus",
"Theophilus", "Keje" };
  var firstFour = people[0..4];

  foreach (var person in firstFour)
  {
    Console.WriteLine(person);
  }
}
```




The range can also be open-ended. That means you can omit the start index, end index or both.

```
var all = people[..]; // contains all the elements from the origin array
var firstFour = people[..4]; // contains "Jane", "Jean", "Grey", and
"Marcus"
var lastTwo = people[4..]; // contains "Theophilus" and "Keje"
```

The `all` variable is open-ended on both ends and therefore returns all the elements. It can also be written as `var all = people[0..people.Length]`. If you omit the start index, it'll use `0` as the start index, and if it's the end index, it'll use the value `sequence.Length` to resolve the value.

You can also declare range variables:

```
Range firstFourRange = ..4
var firstFour = people[firstFourRange]; // contains "Jane", "Jean",
"Grey", and "Marcus"
```

With C# **8**, you can specify that an index is relative to the end of the array. You do this using the `^` operator. Given the array `people`, we can get the last element in the sequence using `people[^1]`. The `^0` is the same as `people.Length`. So if you use `people[^0]` to get the last element of the sequence, you'll get an exception because it's outside the allowed range. We can use this with the range operator:

```
Range lastTwoElement = ^2..
var lastTwo = people[lastTwoElement] // contains "Theophilus" and "Keje"
```

This will give us the last two names. Omitting the end index translates to using `^0` (i.e. `people.Length`) as the end index. We can also assign the index to a variable:




This language support is based on two new types, System.Index and System.Range. The `Index` type represents an index into a sequence, and the `Range` type specifies a sub-range of a sequence.