# Right-Associative Extension Methods: Details



The most general form of leading parameters of an extension method is as follows:

- A possibly empty list of using clauses `leadingUsing`
- A single parameter `extensionParam`
- A possibly empty list of using clauses `trailingUsing`

This is then followed by `def`, the method name, and possibly further parameters `otherParams`. An example is:

```
extension (using a: A, b: B)(using c: C)    // <-- leadingUsing
           (x: X)                              // <-- extensionParam
           (using d: D)                        // <-- trailingUsing
    def +:: (y: Y)(using e: E)(z: Z)           // <-- otherParams
```

An extension method is treated as a right-associative operator (as in SLS §6.12.3) if it has a name ending in `:` and is immediately followed by a single parameter. In the example above, that parameter is `(y: Y)`.

The Scala compiler pre-processes a right-associative infix operation such as `x +: xs` to `xs.+:(x)` if `x` is a pure expression or a call-by-name parameter and to `val y = x; xs.+:(y)` otherwise. This is necessary since a regular right-associative infix method is defined in the class of its right operand. To make up for this swap, the expansion of right-associative extension methods performs an analogous parameter swap. More precisely, if `otherParams` consists of a single parameter `rightParam` followed by `remaining`, the total parameter sequence of the extension method's expansion is:

```
    leadingUsing rightParam trailingUsing extensionParam remaining
```

```
    leadingUsing  rightParam  trailingUsing  extensionParam  remaining
```

For instance, the `+::` method above would become

```
<extension> def +:: (using a: A, b: B)(using c: C)
                    (y: Y)
                    (using d: D)
                    (x: X)
                    (using e: E)(z: Z)
```

This expansion has to be kept in mind when writing right-associative extension methods with inter-parameter dependencies.

An overall simpler design could be obtained if right-associative operators could *only* be defined as extension methods, and would be disallowed as normal methods. In that case neither arguments nor parameters would have to be swapped. Future versions of Scala should strive to achieve this simplification.

‹ Extensi... Imple... ›