TOUR OF SCALA

BY-NAME PARAMETERS

By-name parameters are evaluated every time they are used. They won't be evaluated at all if they are unused. This is similar to replacing the by-name parameters with the passed expressions. They are in contrast to *by-value parameters*. To make a parameter called by-name, simply prepend `=>` to its type.

```
def calculate(input: => Int) = input * 37
```

By-name parameters have the advantage that they are not evaluated if they aren't used in the function body. On the other hand, by-value parameters have the advantage that they are evaluated only once.

Here's an example of how we could implement a while loop:

```
def whileLoop(condition: => Boolean)(body: => Unit): Unit =
  if (condition) {
    body
    whileLoop(condition)(body)
  }

var i = 2

whileLoop (i > 0) {
  println(i)
  i -= 1
}  // prints 2 1
```

The method `whileLoop` uses multiple parameter lists to take a condition and a body of the loop. If the `condition` is true, the `body` is executed and then a recursive call to whileLoop is made. If the `condition` is false, the body is never evaluated because we prepended `=>` to the type of `body`.

Now when we pass `i > 0` as our `condition` and `println(i); i-= 1` as the `body`, it behaves like the standard while loop in many languages.

This ability to delay evaluation of a parameter until it is used can help performance if the parameter is computationally intensive to evaluate or a longer-running block of code such as fetching a URL.

← previous next →

Contributors to this page:

ckipp01 mlachkar yxliang01 ashawley qbahers heathermiller