Scala 3 Reference / New Types / Dependent Function Types

LEARN INSTALL PLAYGROUND FIND A LIBRARY COMMUNITY BLOG

Dependent Function Types

Edit this page on GitHub

A dependent function type is a function type whose result depends on the function's parameters. For example:

```scala
trait Entry { type Key; val key: Key }

def extractKey(e: Entry): e.Key = e.key          // a dependent method

val extractor: (e: Entry) => e.Key = extractKey  // a dependent function value
// ^^^^^^^^^^^^^^^^^^^
// a dependent function type
```

Scala already has *dependent methods*, i.e. methods where the result type refers to some of the parameters of the method. Method `extractKey` is an example. Its result type, `e.Key` refers to its parameter `e` (we also say, `e.Key` *depends* on `e`). But so far it was not possible to turn such methods into function values, so that they can be passed as parameters to other functions, or returned as results. Dependent methods could not be turned into functions simply because there was no type that could describe them.

In Scala 3 this is now possible. The type of the `extractor` value above is

```scala
(e: Entry) => e.Key
```

This type describes function values that take any argument `e` of type `Entry` and return a result of type `e.Key`.

Recall that a normal function type `A => B` is represented as an instance of the `Function1` trait (i.e. `Function1[A, B]`) and analogously for functions with more parameters. Dependent functions are also represented as instances of these traits, but they get an additional refinement. In fact, the dependent function type above is just syntactic sugar for

```
Function1[Entry, Entry#Key]:
  def apply(e: Entry): e.Key
```

More details

‹ Match ... Depen... ›

Contributors to this page

duanebester OlivierBlanvillain robstoll soronpo odersky pikinier20 BarkingBad julienrf michelou b-studios LPTK dvirf1 TheElectronWill sideeffffect

Scaladoc