docs.scala-lang.org

Self-type

2 minutes

Self-types are a way to declare that a trait must be mixed into another trait, even though it doesn't directly extend it. That makes the members of the dependency available without imports.

A self-type is a way to narrow the type of `this` or another identifier that aliases `this`. The syntax looks like normal function syntax but means something entirely different.

To use a self-type in a trait, write an identifier, the type of another trait to mix in, and a `=>` (e.g. `someIdentifier: SomeOtherTrait =>`).

```
trait User {
  def username: String
}

trait Tweeter {
  this: User =>  // reassign this
  def tweet(tweetText: String) =
println(s"$username: $tweetText")
}

class VerifiedTweeter(val username_ : String)
extends Tweeter with User {  // We mixin User
because Tweeter required it
  def username = s"real $username_"
}

val realBeyoncé = new VerifiedTweeter("Beyoncé")
realBeyoncé.tweet("Just spilled my glass of
lemonade")  // prints "real Beyoncé: Just spilled
my glass of lemonade"
```

Because we said `this: User =>` in `trait Tweeter`, now the variable `username` is in scope for the `tweet` method. This also means that since `VerifiedTweeter` extends `Tweeter`, it must also mix-in `User` (using `with User`).

Contributors to this page: