REFINEMENT TYPES

Martin Lechner - 26.07.2018

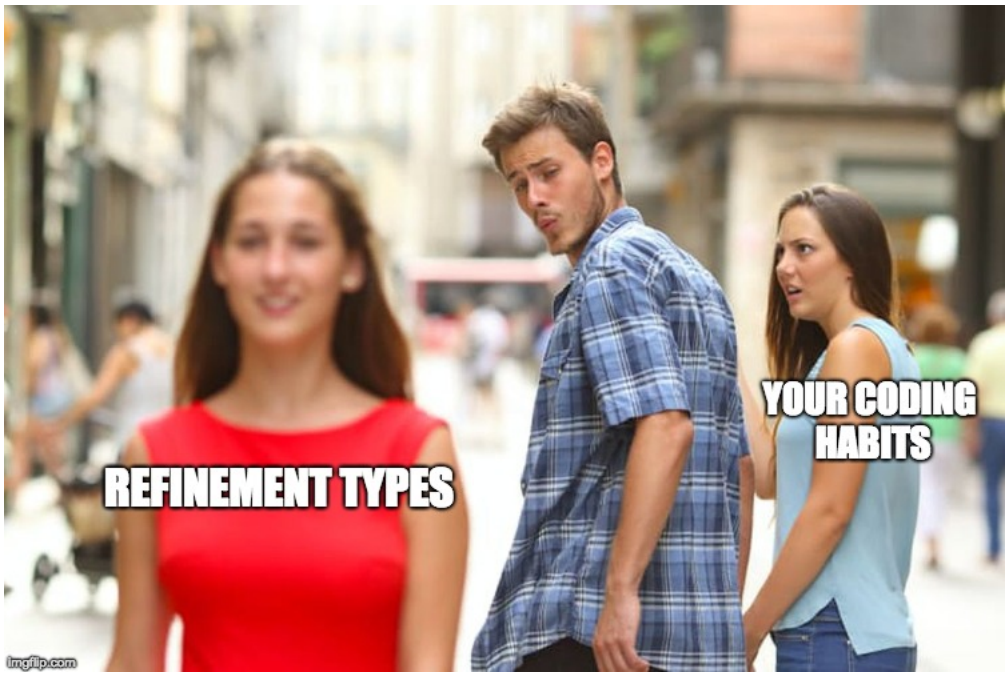

ONE DOES NOT SIMPY
USE LIST[STRING]
imgflip.com


I DON'T ALWAYS LEVERAGE
ADVANCED SCALA
BUT WHEN I DO,
I USE REFINEMENT TYPES
imgflip.com

```scala
import eu.timepit.refined._
// plus some more imports

// Mode 1: Predefined Predicates via Macros
val i: Int Refined Positive = 42

val i: Int Refined Positive = -42
// ⇒ Compile error

// Mode 2: This is the same as 1, but other syntax
refineMV[Positive](42)

// Mode 3: Value not known at compile time → most useful
refineV[Positive](getNumberFromSomewhere())
// ⇒ Either[String, Int Refined Positive]
```

```scala
// Custom refinements → This will be nicer with scala 3's
// literal types
val i: Int Refined Greater[W.`41`.T] = 42
type oneToHundred = Int Refined Interval.ClosedOpen[W.`0`.T, W.`100`.T]
type UUIDString   = String Refined Uuid
type UrlString    = String Refined Url
type EmailString  = String Refined MatchesRegex[W.`".+@.+"`.T]
```

```scala
// other imports
import io.circe.refined._
sealed trait Environment
case object LiveEnvironment
case object OtherEnvironment

implicit val decodeEnvironment: Decoder[Environment] = (c: HCursor) ⇒
  for {
    environment ← c.value.as[NonEmptyString].map(_.value.toLowerCase)
  } yield
  environment match {
    case "live" ⇒ LiveEnvironment
    case _      ⇒ OtherEnvironment
  }
```


REFINEMENT TYPES
YOUR CODING HABITS
imgflip.com

Links

- Github Repo of Scala Refined
- Theory
- Slides
- Slides created with Spectacle!