Scala.meta: the past, the present, and the future

Eugene Burmako (@xeno_by)

12 November 2016

What is scala.meta?

- Implementation-independent metaprogramming library
- Aspiring successor for scala.reflect
- Software product with a stable version and a vibrant community

In today's talk

- Recently released scala.meta 1.x
- Prominent use cases including new-style macro annotations
- Plans for scala.meta 2.x

Scala.meta 1.x

Supported functionality

- Vendor-neutral tree interchange format
- High-fidelity parsing
- First-class tokens
- Integration with Dotty

Parsing: easy to get started

```
scala> import scala.meta._
import scala.meta._

scala> "x + y".parse[Term]
res0: scala.meta.parsers.Parsed[scala.meta.Term] = x + y
```

Parsing: remember all syntactic details

```
scala> import scala.meta._
import scala.meta._

scala> "x + y".parse[Term]
res0: scala.meta.parsers.Parsed[scala.meta.Term] = x + y

scala> "x + y // hello world".parse[Term]
res1: scala.meta.parsers.Parsed[scala.meta.Term] =
x + y // hello world
```

Tokens: remember all syntactic details

```
scala> val add = "x + y // hello world".parse[Term].get
add: scala.meta.Term = x + y // hello world

scala> add.tokens
res2: scala.meta.tokens.Tokens =
Tokens(, x,  , +,  , y,  , // hello world, )
```

Tokens: remember all syntactic details

```
scala> val add = "x + y // hello world".parse[Term].get
add: scala.meta.Term = x + y // hello world

scala> add.tokens
res2: scala.meta.tokens.Tokens =
Tokens(, x,  , +,  , y,  , // hello world, )

scala> add.tokens.structure
res3: String = Tokens(BOF [0..0), x [0..1),  [1..2),
+ [2..3), [3..4), y [4..5),   [5..6),
// hello world [6..20), EOF [20..20))
```

Parsing: support for dialects

```
scala> val sbtBuild = new File(".../project/plugins.sbt")
sbtBuild: java.io.File = .../project/plugins.sbt
```

Parsing: support for dialects

```
scala> val sbtBuild = new File(".../project/plugins.sbt")
sbtBuild: java.io.File = .../project/plugins.sbt

scala> scala.meta.dialects.Sbt0136(sbtBuild).parse[Source]
res4: scala.meta.parsers.Parsed[scala.meta.Source] =

addSbtPlugin("com.typesafe.sbt" % "sbt-pgp" % "0.8.1")

addSbtPlugin("com.eed3si9n" % "sbt-assembly" % "0.11.2")
...
```

Parsing: support for dialects

```
scala> import scala.meta.dialects.Dotty
import scala.meta.dialects.Dotty

scala> "trait Foo(bar: Int)".parse[Source].get
res5: scala.meta.Source = trait Foo(bar: Int)

scala> "Foo & Bar".parse[Type].get
res6: scala.meta.Type = Foo & Bar

scala> res6.structure
res7: String = Type.And(Type.Name("Foo"), Type.Name("Bar"))
```

Quasiquotes: stealing better parts of scala.reflect

```
scala> q"x + y"
res8: scala.meta.Term.ApplyInfix = x + y

scala> val q"$a + $b" = res5
a: scala.meta.Term = x
b: scala.meta.Term.Arg = y
```

Use cases

Use cases

Innovative tooling enabled by scala.meta 1.x:

- Codacy
- Scalafmt
- Scalafix
- New-style macros
- ...

Live demo

Platform-independent new-style macro annotations

Live demo

Platform-independent new-style macro annotations

Try yourself at https://github.com/scalameta/sbt-macro-example

Scala.meta 2.x

Under heavy development

Semantic API:

- Typechecking
- Name resolution
- Type inference
- ...

Semantic API

```
scala> implicit val m = Mirror("...classpath...")
m: Mirror = ...

scala> q"List".tpe
res1: Type = List.type

scala> q"List".defn
res2: Member.Term =
object List extends SeqFactory[List] with Serializable  ...
```

Use cases

- New-style def macros
- Powerful code migrations with Scalafix
- ...

Summary

Summary

- Scala.meta is a thing
- You can use it as a library to write next-gen tooling
- You can use it to write new-style macro annotations

Summary

- Scala.meta is a thing
- You can use it as a library to write next-gen tooling
- You can use it to write new-style macro annotations

- The future will bring even more goodies
- Check out our talks at Scala eXchange 2016
  - “A new macro system for Scala” (Eugene Burmako)
  - “Smooth migrations to Dotty with scalafix ” (Ólafur Páll Geirsson)