

Metaprogramming 2.0

Eugene Burmako (@xeno_by)



ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

11 May 2016

scala.meta is a dream

Easy Metaprogramming For Everyone!

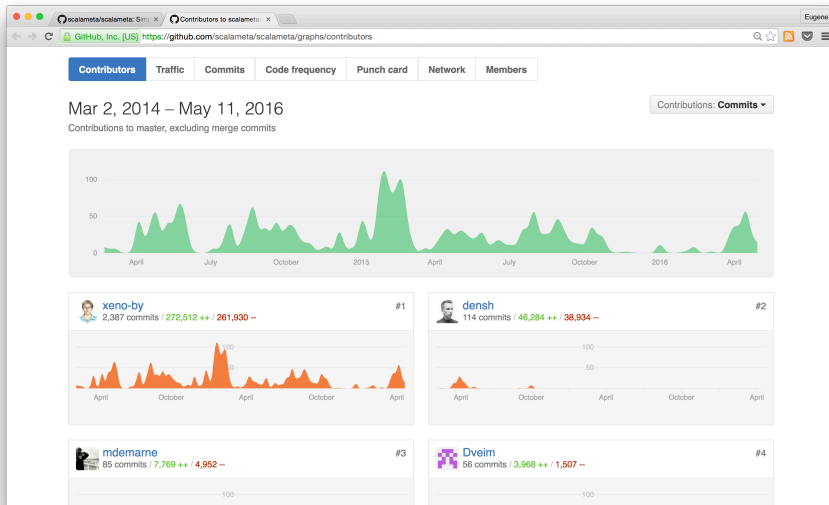
Eugene Burmako (@xeno_by)

Denys Shabalin (@den_sh)

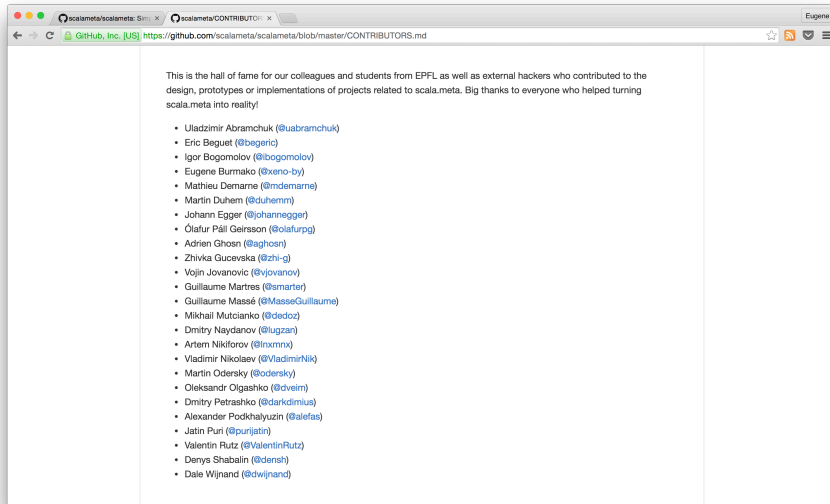
École Polytechnique Fédérale de Lausanne
<http://scalameta.org/>

17 June 2014

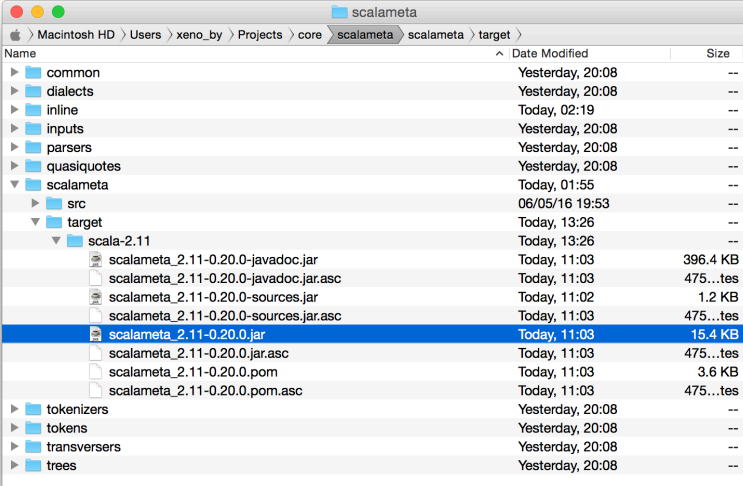
scala.meta is an active project



scala.meta is a community



scala.meta is a product



Name	Date Modified	Size
common	Yesterday, 20:08	--
dialects	Yesterday, 20:08	--
inline	Today, 02:19	--
inputs	Yesterday, 20:08	--
parsers	Yesterday, 20:08	--
quasiquotes	Yesterday, 20:08	--
scalameta	Today, 01:55	--
src	06/05/16 19:53	--
target	Today, 13:26	--
scala-2.11	Today, 13:26	--
scalameta_2.11-0.20.0-javadoc.jar	Today, 11:03	396.4 KB
scalameta_2.11-0.20.0-javadoc.jar.asc	Today, 11:03	475...tes
scalameta_2.11-0.20.0-sources.jar	Today, 11:02	1.2 KB
scalameta_2.11-0.20.0-sources.jar.asc	Today, 11:03	475...tes
scalameta_2.11-0.20.0.jar	Today, 11:03	15.4 KB
scalameta_2.11-0.20.0.jar.asc	Today, 11:03	475...tes
scalameta_2.11-0.20.0.pom	Today, 11:03	3.6 KB
scalameta_2.11-0.20.0.pom.asc	Today, 11:03	475...tes
tokenizers	Yesterday, 20:08	--
tokens	Yesterday, 20:08	--
transversers	Yesterday, 20:08	--
trees	Yesterday, 20:08	--

scala.meta is officially endorsed



Martin Odersky
@odersky



Following

Slides of my Scala Days NYC talk:

Planned In Future Releases

scala.meta

Implicit
Function Types

```
inline def m(inline x: Int, y: Float): Float =  
  meta { ... }
```

Between

- **inline** for inlining, **meta** for meta-programming.
- run by an interpreter (no reflection)
- **meta** uses quasi quotes for matching and construction
- blackbox and annotation macros

ety

Why the change?

- Simpler
- Fewer implementation dependencies
- Safer, since interpretation allows sandboxing
- Restrict syntactic freedom, since no whitebox macros.

50 of 63

Part 1: What can scala.meta do?

ScalaDays 2015

- ▶ Experimentation's temporarily on hold, were now pushing for 0.1
- ▶ Main focus of 0.1 is making scala.meta trees publicly available

ScalaDays 2016

- ▶ 3k commits and 19 milestones later, we're almost there
- ▶ Today we have published our first beta release: v0.20.0
- ▶ We hope v1.0.0 to follow in the near future

Supported functionality

Feature-complete for v1.0.0:

- ▶ Parsing
- ▶ Quasiquotes
- ▶ Tokenization
- ▶ Prettyprinting

Future releases

Planned for v2.0.0:

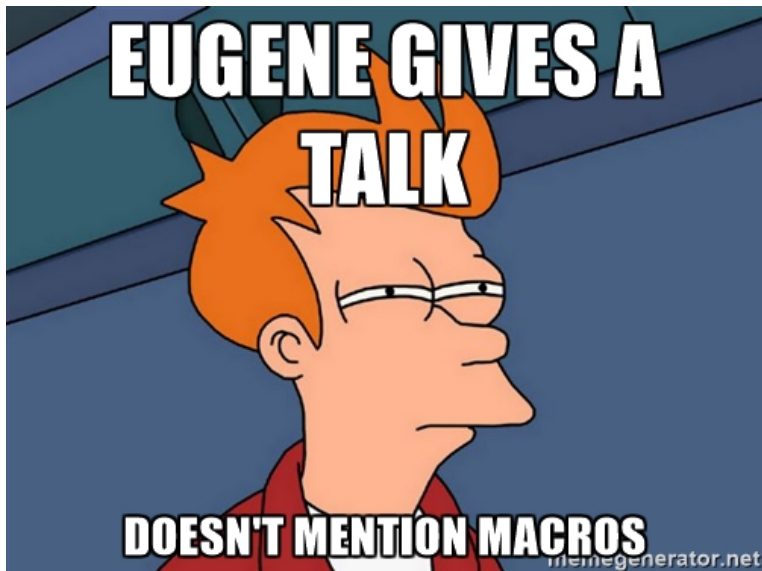
- ▶ AST persistence
- ▶ Name resolution
- ▶ Typechecking
- ▶ Implicit inference

Part 2: Live demo

Summary

- ▶ Parsing different dialects of Scala
- ▶ Type-safe quasiquotes
- ▶ Tokenization for advanced tooling
- ▶ Prettyprinting that respects formatting and comments

Part 3: But what about macros?



Macros are dead



Martin Odersky

@odersky



Following

Slides of my Scala Days NYC talk:

Dropped Features

Procedure Syntax

General Type Projection

Macros

DelayedInit

(the reflection based kind)

```
def m(...) =  
  macro impl(...)
```

Early Initializers

Existential Types

30 of 63

in

Macros are bad

- ▶ Hard to write
- ▶ Don't work with tools well
- ▶ Hopelessly entangled with scalac

Macros are great

- ▶ Enable unique use cases
- ▶ Frequently used by library authors
- ▶ Most of the complexity is incidental

Long live macros



The new future for macros

The screenshot shows a web browser window displaying a GitHub repository page. The repository is 'scalameta/sips', forked from 'scala/scala.github.com'. The current view is a public draft of a file named 'sip-inlinemeta.md' on the 'master' branch. The draft was created by user 'xeno-by' 3 hours ago. The file is 191 lines long, 146 sloc, and 9.81 KB. The draft content includes a table with two columns: 'layout' and 'title'. The table has one row with 'sip' in the 'layout' column and 'Inline Definitions and Meta Expressions' in the 'title' column. Below the table, the draft lists the authors: Eugene Burmako, Sébastien Doeraene, Vojin Jovanovic, Martin Odersky, Dmitry Petrashko, and Denys Shabalin, dated May 2016. The draft title is 'SIP NN: Inline Definitions and Meta Expressions (Public Draft)'.

scalameta / sips
forked from scala/scala.github.com

Unwatch 2 Star 0 Fork 249

Code Pull requests 0 Pulse Graphs Settings

Branch: master sips / sips / drafts / sip-inlinemeta.md Find file Copy path

xeno-by the first public draft of the inline/meta SIP 3b8056d 3 hours ago

1 contributor

191 lines (146 sloc) 9.81 KB Raw Blame History

layout	title
sip	Inline Definitions and Meta Expressions

Eugene Burmako Sébastien Doeraene Vojin Jovanovic Martin Odersky Dmitry Petrashko Denys Shabalin

May 2016

SIP NN: Inline Definitions and Meta Expressions (Public Draft)

The new future for macros

- ▶ SIP NN: Inline Definitions and Meta Expressions
- ▶ We got to the essence of macros and found two orthogonal concepts
- ▶ `inline` for inlining and `meta` for metaprogramming

Old macros

```
class Table[T](val query: Query[T]) {  
  def map[U](fn: T => U): Table[U] = macro Macros.map  
}  
  
object Macros {  
  def map(c: Context)(fn: c.Tree): c.Tree = {  
    val subquery: c.Tree = translate(fn)  
    q"new Table(Map(${c.prefix}, $subquery))"  
  }  
}
```

Old macros

```
val users: Table[User] = ...  
users.map(u => u.name)
```



```
val users: Table[User] = ...  
new Table(Map(users, Ref("name", classOf[String])))
```

- ▶ When the user writes `Table.map`, the compiler calls `Macros.map`
- ▶ `Macros.map` expands in a domain-specific fashion
- ▶ Compiler replaces the call to `Table.map` with the macro expansion

Old macros

```
val users: Table[User] = ...  
users.map(u => u.name)
```



```
val users: Table[User] = ...  
new Table(Map(users, Ref("name", classOf[String])))
```

- ▶ When the user writes `Table.map`, the compiler calls `Macros.map`
- ▶ `Macros.map` expands in a domain-specific fashion
- ▶ Compiler replaces the call to `Table.map` with the macro expansion

Old macros

```
val users: Table[User] = ...  
users.map(u => u.name)
```



```
val users: Table[User] = ...  
new Table(Map(users, Ref("name", classOf[String])))
```

- ▶ When the user writes `Table.map`, the compiler calls `Macros.map`
- ▶ `Macros.map` expands in a domain-specific fashion
- ▶ Compiler replaces the call to `Table.map` with the macro expansion

New macros

```
class Table[T](val query: Query[T]) {  
  inline def map[U](fn: T => U): Table[U] = meta {  
    val subquery: scala.meta.Tree = translate(fn)  
    q"new Table(Map($this, $subquery))"  
  }  
}
```

New macros

```
val users: Table[User] = ...  
users.map(u => u.name)
```



```
val users: Table[User] = ...  
meta{ ...; q"new Table(Map(users, $subquery))" }
```



```
val users: Table[User] = ...  
new Table(Map(users, Ref("name", classOf[String])))
```

- ▶ When the user writes `Table.map`, the compiler inlines its rhs
- ▶ Compiler expands the `meta` block using `scala.meta`
- ▶ The results of the meta expansion are inlined again

New macros

```
val users: Table[User] = ...  
users.map(u => u.name)
```



```
val users: Table[User] = ...  
meta{ ...; q"new Table(Map(users, $subquery))" }
```



```
val users: Table[User] = ...  
new Table(Map(users, Ref("name", classOf[String])))
```

- ▶ When the user writes `Table.map`, the compiler inlines its rhs
- ▶ Compiler expands the meta block using `scala.meta`
- ▶ The results of the meta expansion are inlined again

New macros

```
val users: Table[User] = ...  
users.map(u => u.name)
```



```
val users: Table[User] = ...  
meta{ ...; q"new Table(Map(users, $subquery))" }
```



```
val users: Table[User] = ...  
new Table(Map(users, Ref("name", classOf[String])))
```

- ▶ When the user writes `Table.map`, the compiler inlines its rhs
- ▶ Compiler expands the `meta` block using `scala.meta`
- ▶ The results of the meta expansion are inlined again

Part 4: Live demo

Challenge accepted



Martin Odersky Retweeted



Eugene Burmako @xeno_by · May 10

Lured @odersky to my talk with promise to implement new macro annots. Follow [github.com/scalameta/para...](https://github.com/scalameta/paradise) to see if I can pull it off #scaladays



scalameta/paradise

paradise - <http://event.scaladays.org/scaladays-nyc-2016#!#schedulePopupExtras-7540>

github.com



Mission accomplished



Eugene Burmako @xeno_by · 13h

. @odersky Mission accomplished! Check out [gitter.im/scalameta/scal...](https://gitter.im/scalameta/scalameta) ... for a code sample.



scalameta/scalameta

Simple, robust and portable metaprogramming toolkit for Scala

gitter.im



6



22



In the meanwhile, in the Dotty land

The screenshot shows a GitHub pull request interface. At the top, the browser address bar displays the URL `https://github.com/lampepfl/dotty/pull/1249`. The repository name `lampepfl / dotty` is visible, along with buttons for `Unwatch` (137), `Star` (897), and `Fork` (115). Navigation tabs include `Code`, `Issues` (122), `Pull requests` (19), `Wiki`, `Pulse`, `Graphs`, and `Settings`.

The pull request title is `Evaluate annotations before completing tree of definitions` with the number `#1249`. A green `Open` button is present. Below the title, it states: `odersky wants to merge 1 commit into lampepfl:master from dotty-staging:change-early-annots`. Summary statistics show `Conversation` (0), `Commits` (1), and `Files changed` (1). A small progress bar indicates `+1 -1` changes.

A comment from `odersky` (commented 21 hours ago) is shown. The comment text is: `Motive: That way we can identify annotation macros without special name resolution rules.` and `This was surprisingly easy.` The comment is reviewed by `@xeno-by`. The user's profile information shows they are a `Programming Methods Laboratory EPFL member`.

On the right side, there are sections for `Labels` (None yet), `Milestone` (No milestone), and `Assignee` (No one—assign yourself). A `Notifications` section at the bottom right shows an `Unsubscribe` button and a message: `You're receiving notifications because you were mentioned.`

At the bottom, a green checkmark icon indicates that `All checks have passed` (4 successful checks). A `Show all checks` link is provided. The footer of the browser window shows the URL `https://github.com/lampepfl/dotty/pull/1249/files`.

Part 5: The future

Pattern Creator (Beta)

Scala

New Scala pattern


Gists

Apply to project

View AST

Documentation

Test source	Pattern code
<pre> 1 //Patterns: Custom_Scala_GetCalls 2 package docs.tests 3 4 class Get { 5 6 val optional1: Option[Int] = Option(1) 7 val optional2 = Option(1) 8 val optional3 = None 9 10 //Warn: Custom_Scala_GetCalls 11 optional1.get 12 //Warn: Custom_Scala_GetCalls 13 optional3.get 14 //Warn: Custom_Scala_GetCalls 15 optional1.getOrElse(2) 16 optional2.getOrElse(3) 17 optional3.getOrElse(4) 18 19 //Warn: Custom_Scala_GetCalls 20 Option(1).get 21 22 //Warn: Custom_Scala_GetCalls 23 Option.empty[String].map(identity).get 24 25 //Warn: Custom_Scala_GetCalls 26 None.get 27 28 //Warn: Custom_Scala_GetCalls 29 Some(322).get 30 31 Option(423423).map(_ + 342).getOrElse(432) 32 33 34 </pre>	<pre> 1 import codacy.scalaMetaParser._ 2 import scala.meta._ 3 4 case object Custom_Scala_GetCalls extends Pattern{ 5 6 override def apply(tree: Tree, parameters:Option[PatternParameters]) = { 7 8 tree.collect{ 9 case t@q"\$something.get" if ! t.parent.exists(isApply) => 10 (message(t),t) 11 } 12 13 private[this] def message(tree: Tree) = ResultMessage("Usage of get 14 on optional type.") 15 16 private[this] def isApply(tree: Tree): Boolean = tree match{ 17 case q"\$expr(..\$exprs)" => true 18 case q"\$expr[...\$tpesnel]" => true 19 case _ => false 20 } 21 } </pre>

 [13:11:3] Success: Found 8 matches

The screenshot shows a web browser window with the URL <https://olafurpg.github.io/scalafmt/>. The page title is "Scalafmt - code formatter for Scala" with version "0.2.3". A sidebar on the left contains links: "Scalafmt - code form", "Installation", "Code examples", "Configuration", "FAQ / Troubleshooting", and "Changelog". The main content area has a header "Scalafmt - code formatter for Scala" and version "0.2.3". Below this is a quote: "Any style guide written in English is either so brief that it's ambiguous, or so long that no one reads it. -- Bob Nystrom, 'Hardest Program I've Ever Written', Dart, Google." The text explains that Scalafmt turns messy code into readable, idiomatic, and consistent Scala code. It then shows a side-by-side comparison of code formatting. The left column shows unformatted code, and the right column shows the formatted code. At the bottom, it states: "Scalafmt is an opinionated code formatter. The default style should work great out of the box so you can focus on".

Scalafmt - code formatter for Scala

0.2.3

Any style guide written in English is either so brief that it's ambiguous, or so long that no one reads it.

-- Bob Nystrom, "Hardest Program I've Ever Written", Dart, Google.

Scalafmt turns the mess on the left into the (hopefully) readable, idiomatic and consistent Scala code on the right.

```
object FormatMe { List(number) match {
  case head :: Nil if head % 2 == 0 =>
    "number is even"
  case head :: Nil => "number is not
even"
  case Nil => "List is empty" }
function(arg1, arg2(arg3(arg4, arg5,
"arg6"), arg7 + arg8), arg9.select(1,
2, 3, 4, 5, 6)) }
```

```
object FormatMe {
  List(number) match {
    case head :: Nil
      if head % 2 == 0 =>
        "number is even"
    case head :: Nil =>
      "number is not even"
    case Nil => "List is empty"
  }
  function(
    arg1,
    arg2(arg3(arg4, arg5, "arg6"),
      arg7 + arg8),
    arg9.select(1, 2, 3, 4, 5, 6))
  )
}
```

Published using Scalatra

Scalafmt is an opinionated code formatter. The default style should work great out of the box so you can focus on

Inline macros

```
import scala.meta._

object main {
  inline def apply()(defn: Any) = meta {
    val q"object $name  ..$stats " = defn
    val main = q"""
      def main(args: Array[String]): Unit = { ..$stats }
      """
    q"object $name { $main }"
  }
}

@main object Test {
  println("hello world")
}
```

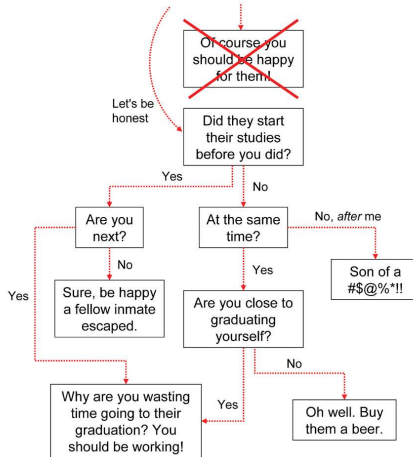
Dotty linker

```
import dotty.linker._

@rewrites
object Rewrites {
  def metaRule[T](x: List[T]) =
    Rewrite(from = x.toString,
             to = meta { /* entry point to scala.meta */ })
}
```

Your friend is graduating.

Should you be happy for them?



WWW.PHDCOMICS.COM

JORGE CUAH © 2013

The road ahead



Eugene Burmako @xeno_by · Mar 7

Totally psyched to announce that this fall I'll be joining Twitter's Engineering Effectiveness group:

[scalamacros.org/news/2016/03/0 ...](http://scalamacros.org/news/2016/03/0...)



58



161



Wrapping up

Summary

- ▶ We've just released our first beta version
- ▶ The project is officially endorsed and funded
- ▶ Current users: Codacy, scalafmt
- ▶ Future users: new macros, Dotty linker
- ▶ Try it out!