

A new macro system for Scala

Eugene Burmako (@xeno_by)



8 December 2016

In today's talk

- ▶ Introduction to Scala macros
- ▶ What's wrong with old-style macros?
- ▶ The design of new-style macros
- ▶ Live demonstration

Introduction to Scala macros

Scala macros

- ▶ Methods or annotations that expand at compile time
- ▶ Logic is written against abstract syntax trees
- ▶ The API is a thin wrapper over Scala compiler internals

@xsd macro annotation

```
@xsd("schema.xsd")
```

```
object schema
```

```
object Test extends App {
```

```
  import schema._
```

```
  val p = new Person("Vassily", "Pupkin")
```

```
  // ...
```

```
}
```

Designing @xsd

```
object schema {  
  class Person(firstName: String, lastName: String)  
}
```

```
object Test extends App {  
  import schema._  
  val p = new Person("Vassily", "Pupkin")  
  // ...  
}
```

Implementing @xsd

Live demonstration

What's wrong with old-style macros?

Problem #1: Hard to write

- ▶ Old-style macros require knowledge of Scala compiler internals
- ▶ Moreover, they are often unnecessarily verbose
- ▶ There is no easy way to troubleshoot errors in macro expansions

Problem #2: Don't work in IntelliJ

- ▶ Old-style macros require Scala compiler internals to run
- ▶ As a result, IntelliJ cannot expand old-style macros
- ▶ Because of the same reason, Dotty cannot expand old-style macros

The design of new-style macros

Key ideas

- ▶ Replace compiler internals with a portable API
- ▶ Make things less verbose while we're at it
- ▶ Implement the API in Scala, IntelliJ, Dotty, etc

Idea #1: Scala.meta

- ▶ Platform-independent metaprogramming library
- ▶ Available for Scala, IntelliJ and Dotty
- ▶ Precisely represents Scala code including whitespace and formatting

Idea #2: Make things less verbose

```
import scala.annotation.StaticAnnotation
import scala.reflect.macros.whitebox.Context

class xsd extends StaticAnnotation {
  def macroTransform(annottees: Any*): Any =
    macro xsdMacro.impl
}

class xsdMacro(val c: Context) {
  def impl(annottees: c.Tree*): c.Tree = {
    import c.universe._
    ...
  }
}
```

Idea #2: Make things less verbose

```
import scala.annotation.StaticAnnotation
import scala.meta._

class xsd extends StaticAnnotation {
  def macroTransform(annottees: Any*): Any =
    macro xsdMacro.impl
}

class xsdMacro {
  def impl(annottees: Tree*): Tree = {
    ...
  }
}
```

Idea #2: Make things less verbose

```
import scala.annotation.StaticAnnotation
import scala.meta._

class xsd extends StaticAnnotation {
  inline def macroTransform(annottees: Any): Any = meta {
    ...
  }
}
```


Idea #2: Make things less verbose

```
import scala.annotation.StaticAnnotation
import scala.meta._

class xsd extends StaticAnnotation {
  inline def apply(defn: Any): Any = meta {
    ...
  }
}
```

Idea #3: Cross-platform design

- ▶ From day one, we've been in touch with the team at JetBrains
- ▶ From day one, we've been thinking about a Dotty implementation

SIP-28/SIP-29 Inline/meta

- ▶ New-style macros are designed via the Scala Improvement Process
- ▶ You can find many more details in the official proposal
- ▶ Join the SIP meetings to get updates about our progress

Live demonstration

Summary

Summary

- ▶ Old-style macros are hard to write and don't work in IntelliJ
- ▶ That's why they are going to be removed from Scala 3
- ▶ We're designing new-style macros via the Scala Improvement Process
- ▶ There is a beta-quality implementation of macro annotations
- ▶ Def macros are next on the todo list

Deliverables

- ▶ Scalac: scalameta/paradise 3.0.0-beta4
- ▶ IntelliJ: JetBrains/intellij-scala 2016.3.2.81
- ▶ Dotty: liufengyun/eden 0.1.2-SNAPSHOT
- ▶ Practical introduction into scala.meta and new-style macros:
<http://scalameta.org/tutorial/>

Credits

- ▶ Design: Denys Shabalin, Martin Odersky et al.
- ▶ Scalac integration: Ólafur Páll Geirsson, Oleksandr Olgashko et al.
- ▶ IntelliJ integration: Mikhail Mutcianko
- ▶ Dotty integration: Martin Odersky, Liu Fengyun