TASTy Inspection


```
libraryDependencies += "org.scala-lang" %% "scala3-tasty-inspector" % scalaVers
```

TASTy files contain the full typed tree of a class including source positions and documentation. This is ideal for tools that analyze or extract semantic information from the code. To avoid the hassle of working directly with the TASTy file we provide the `Inspector` which loads the contents and exposes it through the TASTy reflect API.

Inspecting TASTy files

To inspect the trees of a TASTy file a consumer can be defined in the following way.

```
import scala.quoted.*
import scala.tasty.inspector.*

class MyInspector extends Inspector:
  def inspect(using Quotes)(tastys: List[Tasty[quotes.type]]): Unit =
    import quotes.reflect.*
    for tasty <- tastys do
      val tree = tasty.ast
      // Do something with the tree
```

Then the consumer can be instantiated with the following code to get the tree of the `foo/Bar.tasty` file.

```
object Test:
  def main(args: Array[String]): Unit =
    val tastyFiles = List("foo/Bar.tasty")
    TastyInspector.inspectTastyFiles(tastyFiles)(new MyInspector)
```

Note that if we need to run the main (in the example below defined in an object called `Test`) after compilation we need to make the compiler available to the runtime:

```
scalac -d out Test.scala
scala -with-compiler -classpath out Test
```

Template project

Using sbt version `1.1.5+`, do:

```
sbt new scala/scala3-tasty-inspector.g8
```

in the folder where you want to clone the template.

‹ Reflect... The M... ›