TOUR OF SCALA

ANNOTATIONS

Annotations associate meta-information with definitions. For example, the annotation `@deprecated` before a method causes the compiler to print a warning if the method is used.

```
object DeprecationDemo extends App {
  @deprecated("deprecation message", "release # which deprecates method")
  def hello = "hola"

  hello
}
```

This will compile but the compiler will print a warning: "there was one deprecation warning".

An annotation clause applies to the first definition or declaration following it. More than one annotation clause may precede a definition and declaration. The order in which these clauses are given does not matter.

Annotations that ensure correctness of encodings

Certain annotations will actually cause compilation to fail if a condition(s) is not met. For example, the annotation `@tailrec` ensures that a method is tail-recursive. Tail-recursion can keep memory requirements constant. Here's how it's used in a method which calculates the factorial:

```
import scala.annotation.tailrec

def factorial(x: Int): Int = {

  @tailrec
  def factorialHelper(x: Int, accumulator: Int): Int = {
    if (x == 1) accumulator else factorialHelper(x - 1, accumulator * x)
  }
  factorialHelper(x, 1)
}
```

The `factorialHelper` method has the `@tailrec` which ensures the method is indeed tail-recursive. If we were to change the implementation of `factorialHelper` to the following, it would fail:

```
import scala.annotation.tailrec

def factorial(x: Int): Int = {
  @tailrec
  def factorialHelper(x: Int): Int = {
    if (x == 1) 1 else x * factorialHelper(x - 1)
  }
  factorialHelper(x)
}
```

We would get the message "Recursive call not in tail position".

Annotations affecting code generation

Some annotations like `@inline` affect the generated code (i.e. your jar file might have different bytes than if you hadn't used the annotation). Inlining means inserting the code in a method's body at the call site. The resulting bytecode is longer, but hopefully runs faster. Using the annotation `@inline` does not ensure that a method will be inlined, but it will cause the compiler to do it if and only if some heuristics about the size of the generated code are met.

Java Annotations

When writing Scala code which interoperates with Java, there are a few differences in annotation syntax to note. **Note:** Make sure you use the `-target:jvm-1.8` option with Java annotations.

Java has user-defined metadata in the form of annotations. A key feature of annotations is that they rely on specifying name-value pairs to initialize their elements. For instance, if we need an annotation to track the source of some class we might define it as

```
@interface Source {
  public String URL();
  public String mail();
}
```

And then apply it as follows

```
@Source(URL = "https://coders.com/",
        mail = "support@coders.com")
public class MyClass extends TheirClass ...
```

An annotation application in Scala looks like a constructor invocation, for instantiating a Java annotation one has to use named arguments:

```
@Source(URL = "https://coders.com/",
        mail = "support@coders.com")
class MyScalaClass ...
```

This syntax is quite tedious if the annotation contains only one element (without default value) so, by convention, if the name is specified as `value` it can be applied in Java using a constructor-like syntax:

```
@interface SourceURL {
    public String value();
    public String mail() default "";
}
```

And then apply it as follows

```
@SourceURL("https://coders.com/")
public class MyClass extends TheirClass ...
```

In this case, Scala provides the same possibility

```
@SourceURL("https://coders.com/")
class MyScalaClass ...
```

The `mail` element was specified with a default value so we need not explicitly provide a value for it. However, if we need to do it we can not mix-and-match the two styles in Java:

```
@SourceURL(value = "https://coders.com/",
           mail = "support@coders.com")
           mail = "support@coders.com")
public class MyClass extends TheirClass ...
```

Scala provides more flexibility in this respect

```
@SourceURL("https://coders.com/",
           mail = "support@coders.com")
    class MyScalaClass ...
```

← previous next →

Contributors to this page:

ckipp01 mlachkar SethTisue komainu8 ashawley vrinek jatinguptarally heathermiller

DOCUMENTATION
Getting Started
API
Overviews/Guides
Language Specification

DOWNLOAD
Current Version
All versions

COMMUNITY
Community
Mailing Lists
Chat Rooms & More
Libraries and Tools
The Scala Center

CONTRIBUTE
How to help
Report an Issue

SCALA
Blog
Code of Conduct
License

SOCIAL
GitHub
Twitter