Implicits in Practice (Demo)

Nada Amin
EPFL

Tiark Rompf
Oracle Labs & EPFL

Synopsis: Popularized by Scala, implicits are a versatile language feature that are receiving attention from the wider PL community. This demo will present common use cases and programming patterns with implicits in Scala.

Scala's type checking and type inference engine is able to resolve information through user-defined implicits: (1) implicit arguments to methods, which are resolved by type (2) implicit conversions from one type to another through user-defined conversion functions, and (3) implicit classes, which add additional functionality to existing types. Implicits compose well with other features of Scala (types, traits, objects, macros, lexical scope), enabling generic and type-level programming. Common patterns include "pimp my library" (enriching the interface of existing libraries, in particular legacy Java frameworks), type classes and polytypic programming (well known from Haskell), type constraints (e.g. CanBuildFrom in the standard library collections), and materializers (using macros to generate implicit conversions or type class instances on the fly).

This demo will give an overview of practical programming with implicits. We will distill our experience of using implicits in Scala code on a daily basis into a handful of live coding examples that explain the various patterns and use cases.

While implicits are a powerful feature for library designers that can enable concise programming patterns in user code, using them to good effect requires consideration and taste. One criticism that has been levied against them is that programs become harder to understand beyond the superficial level because many things are going on behind the back of the application programmer. Indeed, debugging implicits-heavy code can become complicated, in particular due to unfamiliar and unexpected type errors caused by inapplicable or ambiguous implicits.

Our demo will emphasize common pitfalls of programming with implicits, and how they can be prevented with a little bit of programming discipline. Our demo will further demonstrate how we cope with errors through some live debugging, what hoops we have to and don't have to jump through (e.g. when changing a method to take extra implicit parameters), and how tools help or could help.