

3 Javaとの連携

Scala の長所の一つは Java コードとの連携が非常に簡単だということです。java.lang パッケージの全てのクラスはデフォルトでインポートされます。その他は明示的にインポートする必要があります。

それを示す例をお見せしましょう。現在時刻を取得して特定の国、例えばフランス*1で使われる表記方法を適用したいとしましょう。

Java のクラスライブラリには Date と DateFormat のような強力なユーティリティクラスがあります。Scala は Java とシームレスに連動するので、Scala のクラスライブラリに同様なクラスを実装する必要はありません。対応する Java パッケージのクラスをそのままインポートできます。

```
import java.util.{Date, Locale}
import java.text.DateFormat
import java.text.DateFormat._

object FrenchDate {
  def main(args: Array[String]) {
    val now = new Date
    val df = getInstance(LONG, Locale.FRANCE)
    println(df format now)
  }
}
```

Scala の import 文は Java のものと大変似ていますが、ずっと強力です。同じパッケージの複数のクラスを、一行目のように波カッコ({})で囲むことでいっぺんにインポートできます。他に違うのは、全てのパッケージあるいはクラスの名前をインポートする時には、アスタリスク(*)の代わりにアンダースコア(_)を使うことです。後で示すように、アスタリスクは Scala では有効な識別子(例えばメソッド名)だからです。

従って3行目の import 文は DateFormat クラスの全てのメンバをインポートします。これによって静的メソッド getInstance と静的フィールド LONG を直接見えるようにします。

main メソッドでは最初に、デフォルトで現在時刻を持つ Java の Date クラスのインスタンスを生成します。次に、先にインポートした静的メソッド getInstance を用いて日付フォーマットを定義します。最後に、各国語化された DateFormat インスタンスによってフォーマットされた現在時刻を表示します。最後の行は Scala 構文の興味深い特徴を示しています。引数を一つ取るメソッドには中置構文を使えます。つまりこの

```
df format now
```

という式は、次の式の文字数の少ない表記方法だといえます。

```
df.format(now)
```

これは些細な文法規則に見えるかもしれませんが、実は重要な事です。詳しくは次節で述べます。

Java との統合に関するこの節を終えるにあたって、Scala では直接に Java クラスを継承したり Java のインターフェイスを実装したりできる、という事も述べておくべきでしょう。

*1 例えばスイスでフランス語を話す地域など、他の地方でも同じ記法が使用されます。

名前:	<input type="text"/>
コメント:	<input type="text"/>

投稿