

第 10 章 For 内包表記 (For-Comprehensions)

前の章では map, flatMap, filter のような高階関数がリストを扱う強力な道具となることを示しました。しかし時に、これらの関数が要求する抽象化レベルゆえに、プログラムが理解しにくくなることもあります。

理解しやすくするために、Scala には高階関数適用の共通パターンを簡単にする特別な記法があります。この記法は数学における集合の内包表記と、C や Java のような命令型言語の for ループとの架け橋となるものです。また関係データベースのクエリ(問い合わせ)記法にもよく似ています。

最初の例として、name と age をフィールドにもつ人(person)のリスト persons があるとしましょう。そのリストの中で、年齢が 20 を超える人の名前を表示するのは、次のように書けます。

```
for (p <- persons if p.age > 20) yield p.name
```

これは高階関数 filter と map を使った次の式と等価です。

```
persons filter (p => p.age > 20) map (p => p.name)
```

for 内包表記は命令型言語における for ループにすこし似ていますが、各繰り返しの結果値をすべて集めてリストを作る点が違います。

一般的に、for 内包表記は次の形式です。

```
for ( s ) yield e
```

ここで s は **ジェネレータ**、**定義**、**フィルタ** のシーケンス(並び)です。**ジェネレータ** とは val x <- e という形 (ただし e は値がリストとなる式) です。それによって x はリスト中の値で次々に束縛されます。**定義** とは、val x = e という形です。残りの内包表記において、x は e の値の名前として導入されます。**フィルタ** とは Boolean 型の式 f です。f が false となるような束縛はすべて無視されます。シーケンス s は常にジェネレータで始まります。もしいくつかのジェネレータがシーケンスに含まれる場合、後にあるジェネレータは前にあるものよりも先に変化します。

シーケンス s は丸括弧 () ではなく波括弧 {} で囲むこともでき、その場合はジェネレータ、定義、フィルタの間のセミコロンは省略できます。

for 内包表記を使う例を二つ示します。最初に、前の章の例を繰り返しましょう。与えられた正整数 n に対し、正整数 i と j のすべての組 (ただし $1 \leq j < i < n$ で、i+j が素数) を求めなさい、for 内包表記を使うと、この問題は次のように解けます。

```
for { i <- List.range(1, n)
      j <- List.range(1, i)
      if isPrime(i+j) } yield {i, j}
```

前に考えた map, flatMap, filter を使った解よりずっと明快であるのは間違いないでしょう。

二つ目の例として、2 つのベクトル xs と ys のスカラー積の計算を考えましょう。for 内包表記を使って次のように書けます。

```
sum(for ((x, y) <- xs zip ys) yield x * y)
```

- [10.1 N クイーン問題 \(The N-Queens Problem\)](#)
- [10.2 For 内包表記によるクエリ](#)
- [10.3 For 内包表記の変換](#)
- [10.4 For ループ](#)
- [10.5 For の一般化 \(Generalizing For\)](#)

[前ページ](#) [10 章 目次](#) [次ページ](#)

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

投稿