Programmer en Erlang

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 2 mars 2011

https://www.bortzmeyer.org/programmer-en-erlang.html

Auteur(s) : Francesco Cesarini, Simon Thompson
ISBN n°978-2-7440-2445-0
Éditeur : Pearson
Publié en 2010

Ce livre est un des premiers sur Erlang en français (je n'ai pas lu celui de Mickaël Rémond). Le langage de programmation Erlang occupe une niche originale en étant à la fois un langage fonctionnel et un langage parallèle. Ses promoteurs le disent particulièrement adaptés aux grands systèmes temps-réel complexes, comme les autocommutateurs d'où il vient. Mais Erlang a aussi été utilisé pour des serveurs Internet qu'on peut installer sur ses machines Unix normales, comme la base de données NoSQL CouchDB ou comme le serveur XMPP ejabberd.

D'abord, un mot sur le livre, avant de revenir au langage. [Pour information : j'ai reçu un exemplaire gratuit de l'éditeur, en tant que blogueur supposé influent.] Il est écrit en anglais par deux spécialistes du langage. Ceux-ci ne manquent pas d'humour autocritique comme le montre l'exercice 17.4 « Des deux auteurs du livre, qui est celui qui a réussi à provoquer tout seul une panne d'un réseau mobile au niveau national en utilisant sans précaution l'outil de trace dbg ? ». Leur livre couvre tous les aspects, aussi bien sur le langage lui-même, que sur des questions de plus haut niveau, comme les patrons de conception pour la programmation parallèle, les bonnes pratiques pour la gestion des erreurs, et les détails pratiques (comme le débogueur, cité dans l'exercice ci-dessus) ou les logiciels auxiliaires comme le système de documentation EDoc <http://www.erlang.org/doc/apps/edoc/>. Le livre est très concret, plein d'exemples (qui marchent) et d'exercices. Cela va jusqu'à du code pour décoder les paquets TCP, qui vous donnera une bonne idée des capacités « bas niveau » d'Erlang. Avec Erlang, on fait du "*pattern matching*" même sur les bits entrants. Si vous avez passé une journée à faire de l'ISO 9001, ce livre est un excellent traitement à prendre dans la soirée. Compte-tenu à la fois de l'importance de ce domaine pour Erlang, et de sa relative difficulté, je préviens que la programmation parallèle occupe une grande part du livre.

La traduction est d'Éric Jacoboni, un vieux de la vieille, un expert qui a déjà produit de très nombreuses documentations et livres. (Pour les amateurs de souvenirs, voir sa documentation sur sendmail <http://www.linux-france.org/article/mail/sendmail/>.) Il s'est bien tiré des défis habituels de la traduction de livres techniques en français. Les exemples ont ainsi reçu des noms de variables,

de types et de fonctions en français, ce qui aide à les différencier facilement des mots-clés (Erlang n'a pas de règle pour différencier une fonction définie par l'utilisateur de celles du système). On a donc des « processus ouvriers » (je n'ai pas vérifié dans le texte original, je suppose que c'étaient des "*worker thread*"), des enregistrements et pas des "*records*" et on se lie au lieu de se "*binder*". Certains termes ne sont pas traduits mais toujours pour de bonnes raisons. Ainsi, les "*fun*" (les lambda d'Erlang) gardent leur nom, puisque c'est aussi un mot-clé. Le "*pattern matching*", concept essentiel en Erlang, garde prudemment son nom. Plus curieusement, les compétitions sont restées des "*races*".

Le contenu du livre est globalement de très bonne qualité. La seule erreur que j'ai trouvée est l'amusante bogue en bas de la page 459, où une note du relecteur est restée dans la version imprimée. En revanche, la qualité physique du livre est plus contestable : plusieurs pages ne sont tout simplement pas fixées et partent spontanément. À l'heure où les livres papier sont tant menacés par le numérique, c'est un problème sérieux, s'ils ne sont même pas pratiques à utiliser.

Et le langage lui-même ? Erlang est très déroutant pour celui qui vient des langages impératifs traditionnels comme C. Ainsi, on ne peut lier une valeur qu'à une variable (qui commence par une majuscule) :

```
> A = 4.
4

> a = 4.
** exception error: no match of right hand side value 4
```

C'est logique, mais le message d'erreur est déroutant !

Autre sujet d'étonnement si on a lu trop vite l'excellente section sur le "*pattern matching*" :

```
> {Foo, Bar} = {1, 2}.
{1,2}

> {Foo, Bar} = {3, 4}.
** exception error: no match of right hand side value {3,4}

> {Foo, Bar} = {1, 2}.
{1,2}
```

Comme dans beaucoup de langages fonctionnels, on ne peut en effet affecter une valeur à une variable (on dit **lier**) qu'une fois. Mais Erlang est plus intelligent que ça, on peut affecter plusieurs fois, si c'est à la même valeur (car l'affectation n'est qu'un cas particulier du "*pattern matching*"). C'est donc pour cela que le troisième cas réussit.

Globalement, le langage est très intéressant (je n'écris pas « prometteur » : Erlang est un vieux langage même s'il est encore peu connu), utilisable dès aujourd'hui pour des projets réels et ce livre est un très bon moyen de s'y plonger. Je le recommande à tous les programmeurs, pour élargir leur perspective sur la programmation.

Sur ce créneau de langages de (relativement) bas niveau, avec parallélisme intégré, le principal concurrent d'Erlang est sans doute Go dont j'ai déjà parlé <https://www.bortzmeyer.org/go-langage.html>. Tout le monde n'aime pas forcément Erlang et la critique la plus argumentée que j'ai vue est « "*What Sucks About Erlang*" <http://damienkatz.net/2008/03/what_sucks_abou.html> ». Un bon article de défense d'Erlang est « "*An Open Letter to the Erlang Beginner (or Onlooker)*" <http://ferd.ca/an-open-letter-to-the-erlang-beginner-or-onlooker.html> ».