

28.04.2001 - version 5.5

- Nettoyage du code html [Armel].

14.07.2000 - version 5.4

- Quelques espaces oubliées.

26.06.2000 - version 5.3

- Insertion du journal de log.

18.06.2000 - version 5.2

- Modification des guillemets " " en « ».

14.06.2000 - version 5.1

- Première publication sur eGroups.

D : Ressources

Logicielles

Le JDK de *java.sun.com*. Même si vous choisissez d'utiliser un autre environnement de développement, il est toujours bon d'avoir le JDK sous la main au cas où vous vous heurteriez à ce qui pourrait être une erreur du compilateur. Le JDK est la référence ; et s'il y a un bug, il a de fortes chances d'être rapidement repéré.

La documentation HTML de Java de *java.sun.com*. Je n'ai jamais trouvé de livre de référence sur la bibliothèque standard Java qui soit à jour ou complet. Bien que la documentation HTML de Sun soit parsemée de petits bugs et quelquefois un peu laconique, au moins on y trouve toutes les classes et méthodes. Souvent on se sent moins enclin à utiliser une ressource online plutôt qu'un livre imprimé, mais cela vaut généralement le coup de surmonter cette appréhension et de consulter d'abord la documentation HTML, afin de se faire au moins une idée générale. Si cela ne suffit pas, alors on peut toujours se tourner vers une ressource plus traditionnelle.

Livres

Thinking in Java, 1st Edition. Disponible sur le CD ROM fourni avec ce livre ou téléchargeable gratuitement depuis *www.BruceEckel.com* sous forme HTML pleinement indexée avec colorisation de syntaxe. Contient des informations moins récentes ou jugées pas assez pertinentes pour demeurer dans la deuxième édition.

Core Java 2, de Horstmann & Cornell, Volume I-Fundamentals (Prentice-Hall, 1999). Volume II-Advanced Features, 2000. Enorme, détaillé, et le premier endroit où je vais lorsque je cherche une réponse. Le livre que je recommande une fois terminé *Thinking in Java* et que vos besoins sont plus conséquents.

Java in a Nutshell: A Desktop Quick Reference, 2nd Edition, de David Flanagan (O'Reilly, 1997). Un résumé condensé de la documentation Java online. Personnellement, je préfère parcourir la documentation online de *java.sun.com*, en particulier parce qu'elle est si souvent mise à jour. Cependant, nombreux sont ceux qui préfèrent une documentation imprimée et celle-ci convient parfaitement ; de plus elle contient plus de commentaires que les documents online.

The Java Class Libraries: An Annotated Reference, de Patrick Chan et Rosanna Lee (Addison-Wesley, 1997).

Ce que la référence online *aurait dû* être : posséder assez de descriptions pour être utilisable. Un des relecteurs techniques de *Thinking in Java* a dit : « Si je ne devais avoir qu'un livre sur Java, ce serait celui-ci (en plus du tien, bien sûr) ». Je ne suis pas aussi enthousiaste que lui. Il est gros, il est cher, et la qualité de ses exemples ne me satisfait pas. *Mais* c'est un endroit où regarder quand on est coincé et il semble avoir plus de profondeur (et une taille plus conséquente) que *Java in a Nutshell*.

Java Network Programming, de Eliotte Rusty Harold (O'Reilly, 1997). Je n'ai commencé à comprendre la programmation réseau en Java qu'après avoir lu ce livre. J'ai aussi trouvé que son site web, Café au Lait, offrait un point de vue stimulant, pertinent, à jour et non influencé sur les développements de Java. Ses mises à jour fréquentes suivent l'actualité de Java. Se reporter à metalab.unc.edu/javafaq/.

JDBC Database Access with Java, de Hamilton, Cattell & Fisher (Addison-Wesley, 1997). Si vous n'y connaissez rien en SQL et aux bases de données, ce livre vous en fournira les premiers éléments. Il contient aussi une « référence annotée » ainsi que des détails de l'API (encore une fois, de ce que l'API était lors de la sortie du livre). L'inconvénient de ce livre, comme tous ceux de *The Java Series* (« Les SEULS Livres Approuvés par JavaSoft »), est qu'ils sont édulcorés en ce sens qu'ils ne disent que du bien de Java - vous ne trouverez aucune critique dans cette collection.

Java Programming with CORBA, de Andreas Vogel & Keith Duddy (John Wiley & Sons, 1997). Un sujet sérieusement traité avec des exemples de code pour trois ORBs Java (Visibroker, Orbix, Joe).

Design Patterns, de Gamma, Helm, Johnson & Vlissides (Addison-Wesley, 1995). Le livre qui a lancé le concept de patrons dans la programmation.

Practical Algorithms for Programmers, de Binstock & Rex (Addison-Wesley, 1995). Les algorithmes sont en C, ils sont donc facilement transposables en Java. Chaque exemple est minutieusement expliqué.

Analyse & conception

Extreme Programming Explained, de Kent Beck (Addison-Wesley, 2000). J'adore ce livre. Bien que j'aie tendance à utiliser une approche radicale, j'ai toujours cru qu'il devait exister une manière différente et plus efficace de mener à bien un projet, et je crois que XP s'en approche drôlement. Le seul livre qui m'ait fait autant d'effet est *PeopleWare* (décrit ci-après), qui parle principalement de l'environnement et traite de la culture d'entreprise. *Extreme Programming Explained* parle de programmation, et remet les choses, y compris les dernières « trouvailles », à leur place. Il va même jusqu'à dire que les images sont OK tant qu'on ne passe pas trop de temps avec elles et qu'on les jette à terme (vous remarquerez que ce livre ne porte pas la mention « approuvé par UML » sur sa couverture). J'ai vu des gens choisir une entreprise en se basant seulement sur le fait de savoir si celle-ci utilisait XP. Petit livre, petits chapitres, agréable à lire, profond quant à la réflexion qu'il provoque. Vous vous imaginez dans une telle atmosphère de travail et cela vous amène des visions d'un monde entièrement nouveau.

UML Distilled, 2nd Edition, de Martin Fowler (Addison-Wesley, 2000). La première rencontre avec UML est souvent rebutante à cause de tous les diagrammes et les détails. Selon Fowler, la plupart de ces détails ne sont pas nécessaires, et il tranche dedans pour aller à l'essentiel. Pour la plupart des projets, vous n'avez besoin de connaître que quelques diagrammes outils, et le but de Fowler est d'arriver à une bonne conception plutôt que de s'ennuyer avec tous les détails pour y arriver. Un petit livre agréable et intéressant ; le premier à lire si vous avez besoin de vous plonger dans UML.

UML Toolkit, de Hans-Erik Eriksson & Magnus Penker, (John Wiley & Sons, 1997). Explique UML et comment l'utiliser, avec un exemple appliqué en Java. Un CD ROM contient le code Java du livre et une version allégée de Rational Rose. Une excellente introduction sur UML et comment l'utiliser pour construire un système réel.

The Unified Software Development Process, de Ivar Jacobsen, Grady Booch, et James Rumbaugh (Addison-Wesley, 1999). Je m'étais attendu à détester ce livre. Il semblait avoir tous les défauts d'un texte universitaire ennuyeux. Je fus agréablement surpris - seules quelques parties du livre contiennent des explications qui font penser que les concepts ne sont pas clairs pour les auteurs. La majeure partie du livre n'est pas seulement claire, mais agréable à lire. Et le mieux, c'est que la méthodologie présentée n'est pas dénuée de sens pratique. Ce n'est pas la Programmation Extreme (en particulier elle ne présente pas la même lucidité quant aux tests), mais elle fait partie du mastodonte UML - même si vous n'arrivez pas à faire adopter XP, la plupart des gens ont adhéré au « mouvement UML » (sans tenir compte de leur niveau *réel* d'expérience) et vous devriez pouvoir faire adopter cette méthode. Je pense que ce livre est le vaisseau amiral de UML, et celui que vous devriez lire après *UML Distilled* de Fowler quand vous aurez besoin de plus de détails.

Avant d'opter pour une méthode, il est bon de se renseigner auprès de personnes qui n'essayent pas d'en vendre une. Il est facile d'adopter une méthode sans réellement comprendre ce qu'on en attend ou ce qu'elle va faire pour nous. « D'autres l'utilisent, ce qui fait une bonne raison de l'adopter ». Cependant, les hommes peuvent montrer une bizarrerie psychologique : s'ils pensent que quelque chose va solutionner leurs problèmes, ils l'essayeront (c'est l'expérimentation, ce qui est une bonne chose). Mais si cela ne résoud pas leurs problèmes, ils peuvent redoubler d'efforts et annoncer à voix haute quelle découverte merveilleuse ils ont fait (c'est un mensonge, ce qui est une vilaine chose). Le raisonnement fait ici est que si vous arrivez à attirer d'autres personnes dans le même bateau, vous ne vous retrouverez pas tout seul, même s'il ne va nulle part (ou coule).

Cela ne veut pas dire que toutes les méthodologies se terminent en cul-de-sac, mais que vous devez vous préparer pour vous maintenir dans le mode expérimentation (« Cela ne marche pas, essayons quelque chose d'autre »), sans rentrer dans le mode mensonge (« Non, ce n'est pas réellement un problème. Tout marche, on n'a pas besoin de changer »). Je pense que les livres suivants, à lire *avant* de choisir une méthodologie, vous aident dans cette préparation.

Software Creativity, de Robert Glass (Prentice-Hall, 1995). Ceci est le meilleur livre que j'aie vu qui discute des perspectives du problème de la méthodologie. C'est un recueil de courts essais et articles que Glass a écrit et quelquefois récupéré (P. J. Plauger est l'un des contributeurs), reflétant ses années d'étude et de réflexion sur le sujet. Ils sont distrayants et juste assez longs pour expliquer ce qu'ils veulent faire passer ; il ne vous égare pas ni ne vous ennue. Glass ne vous mène pas en bateau non plus, il y a des centaines de références à d'autres articles et études. Tous les programmeurs et les directeurs devraient lire ce livre avant de s'aventurer dans la jungle des méthodologies.

Software Runaways: Monumental Software Disasters, de Robert Glass (Prentice-Hall, 1997). Le point fort de ce livre est qu'il met en lumière ce dont personne ne parle : combien de projets non seulement échouent, mais échouent spectaculairement. La plupart d'entre nous pensent encore : « Cela ne peut pas m'arriver » (ou « Cela ne peut pas *encore* m'arriver »), et je pense que cela nous met en position de faiblesse. En se rappelant que les choses peuvent toujours mal se passer, on se retrouve en bien meilleure position pour les contrôler.

Peopleware, 2nd Edition, de Tom Demarco et Timothy Lister (Dorset House, 1999). Bien que les auteurs proviennent du monde du développement logiciel, ce livre traite des projets et des équipes en général. Mais il se concentre sur les *gens* et leurs besoins, plutôt que sur la technologie et ses besoins. Les auteurs parlent de la création d'un environnement où les gens seront heureux et productifs, plutôt que des décisions et des règles que ces gens devraient suivre pour être les composants bien huilés d'une machine. C'est cette dernière attitude qui, selon moi, fait sourire et acquiescer les programmeurs quand la méthode XYZ est adoptée et qu'ils continuent à travailler de la même manière qu'ils l'ont toujours fait.

Complexity, de M. Mitchell Waldrop (Simon & Schuster, 1992). Ce livre rapporte la mise en relations d'un groupe de scientifiques de différentes spécialités à Santa Fe, New Mexico, pour discuter de problèmes que leurs disciplines ne pouvaient résoudre individuellement (le marché de la bourse en économie, la création originelle

de la vie en biologie, pourquoi les gens agissent ainsi qu'ils le font en sociologie, etc...). En combinant la physique, l'économie, la chimie, les mathématiques, l'informatique, la sociologie et d'autres matières, une approche multidisciplinaire envers ces problèmes se développe. Mais plus important, une nouvelle façon de *penser* ces problèmes ultra-complexes émerge : loin du déterminisme mathématique et de l'illusion qu'on peut écrire une équation qui prédise tous les comportements, mais basée sur l'*observation* à la recherche d'un motif et la tentative de recréer ce motif par tous les moyens possibles. (Le livre rapporte, par exemple, l'émergence des algorithmes génétiques.) Je pense que cette manière de penser est utile lorsqu'on observe les façons de gérer des projets logiciels de plus en plus complexes.

Python

Learning Python, de Mark Lutz and David Ascher (O'Reilly, 1999). Une bonne introduction à ce qui est rapidement devenu mon langage favori, un excellent compagnon à Java. Le livre inclut une introduction à JPython, qui permet de combiner Java et Python dans un unique programme (l'interpréteur JPython est compilé en pur bytecode Java, il n'y a donc rien de spécial à ajouter pour réaliser cela). Cette union de langages promet de grandes possibilités.

La liste de mes livres

Listés par ordre de publication. Tous ne sont pas actuellement disponibles.

Computer Interfacing with Pascal & C, (Auto-publié dans la rubrique Eisisys, 1988. Disponible seulement sur www.BruceEckel.com). Une introduction à l'électronique au temps où le CP/M était encore le roi et le DOS un parvenu. J'utilisais des langages de haut niveau et souvent le port parallèle de l'ordinateur pour piloter divers projets électroniques. Adapté de mes chroniques dans le premier et le meilleur des magazines pour lesquels j'ai écrit, *Micro Cornucopia* (pour paraphraser Larry O'Brien, éditeur au long cours de *Software Development Magazine* : le meilleur magazine informatique jamais publié - ils avaient même des plans pour construire un robot dans un pot de fleurs !). Hélas, *Micro C* s'est arrêté bien avant que l'Internet n'apparaisse. Créer ce livre fut une expérience de publication extrêmement satisfaisante.

Using C++, (Osborne/McGraw-Hill, 1989). Un des premiers livres sur le C++. Epuisé et remplacé par sa deuxième édition, renommée *C++ Inside & Out*.

C++ Inside & Out, (Osborne/McGraw-Hill, 1993). Comme indiqué, il s'agit en fait de la deuxième édition de ***Using C++***. Le C++ dans ce livre est raisonnablement précis, mais il date de 1992 et *Thinking in C++* est destiné à le remplacer. Vous pouvez trouver plus de renseignements sur ce livre et télécharger le code source sur www.BruceEckel.com.

Thinking in C++, 1st Edition, (Prentice-Hall, 1995).

Thinking in C++, 2nd Edition, Volume 1, (Prentice-Hall, 2000). Téléchargeable sur www.BruceEckel.com.

Black Belt C++, the Master's Collection, Bruce Eckel, éditeur (M&T Books, 1994). Epuisé. Un recueil de chapitres par divers experts C++, basés sur leurs présentations dans le cycle C++ lors de la Conférence sur le Développement Logiciel, que je présidais. La couverture de ce livre m'a convaincu d'obtenir le contrôle de toutes les futures couvertures.

Thinking in Java, 1st Edition, (Prentice-Hall, 1998). La première édition de ce livre a gagné la *Software Development Magazine Productivity Award*, le *Java Developer's Journal Editor's Choice Award*, et le *JavaWorld Reader's Choice Award for best book*. Téléchargeable sur www.BruceEckel.com.