Guide des outils

Sommaire

Thierry Vaira - <tvaira@free.fr> - version v0.3 - 23/08/2021 - btssn-lasalle84.github.io

1. Le métier de développeur logiciel

On nomme « développeurs » les personnes qui conçoivent et mettent à jour les logiciels informatiques. Pour cela, le développeur met en œuvre des langages de programmation et travaille souvent en communautés avec d'autres développeurs.

Un développeur *full stack* est un développeur web capable de réaliser la programmation d'un site ou d'une application web à la fois en *front-end* et *back-end*. Le développement *front-end* correspond à l'interface d'une application qu'un utilisateur peut voir et avec lesquelles il peut interagir directement (par exemple, HTML, CSS et JavaScript pour une page web). Par opposition, le développement *back-end* (la partie cachée de l'iceberg) désigne l'ensemble du code devant produire un résultat (par exemple, PHP pour un serveur produisant une page web).

Comme dans tout métier, le développeur utilise des outils pour l'aider à produire un logiciel de manière efficace. Les outils cités ci-dessous sont plutôt réservés à des experts du domaine et ne conviennent pas à des débutants effectuant leurs premiers apprentissages.

> Le marteau ne fait pas l'architecte !

2. Environnement de développement minimal

2.1. L'éditeur de texte

Au minimum, le développeur utilisera un **éditeur de texte**. Ce logiciel lui permettra de produire du **code source**.

Un éditeur de texte est un logiciel destiné à la création et l'édition de **fichiers textes**. Les fichiers texte sont des fichiers qui contiennent les séquences brutes d'octets qui encodent simplement les caractères. On parle aussi de « texte brut » (*plain text*), par opposition aux logiciels de traitement de texte qui produisent des « documents » texte (contenant par exemple des élements de mise en forme).

La recherche d'un « bon » éditeur de texte est important pour le développeur. Il devra par exemple posséder au minimum ces fonctionnalités :

- le copier/coller
- l'ouverture simultanée de plusieurs fichiers
- la recherche et remplacement de texte ou motifs (notamment leses expressions rationnelles)
- l'indentation automatique
- le complètement automatique (autocomplétion)
- la coloration syntaxique
- des raccourcis clavier, des macros
- la gestion de différents encodages de caractères (unicode ...)

Sous Windows :

- Bloc-notes (Notepad) l'éditeur standard de Windows
- Notepad++, UltraEdit, Sublime Text, …

Sous UNIX - GNU/Linux :

- Vi, Vim
- Emacs
- nano, …

Sous Mac OS, Mac OS X et macOS :

- SimpleText, TextEdit, …

À lire : Guerre d'éditeurs

Pour le respect des règles de codage et bonnes pratiques en C++ : https://github.com/btssn-lasalle84/github-actions-cpp

2.2. Le compilateur

Dans le cas de langages compilés, le développeur aura besoin d'une chaîne de fabrication comprenant notamment un **compilateur**.

Support de cours : gcc/g++

2.3. Le moteur de production

Les étapes de fabrication d'un exécutable peuvent être automatisées en utilisant un outil, comme `make`.

`make` est un logiciel traditionnel d'UNIX. C'est un "**moteur de production**" : il sert à appeler des commandes créant des fichiers.

Liens : https://www.gnu.org/software/make/ et https://www.gnu.org/software/make/manual/html_node/index.html

Support de cours : cours-c-programmation-modulaire.pdf et make

À la différence d'un simple script *shell*, `make` exécute les commandes seulement si elles sont nécessaires. Le but est d'arriver à un résultat (logiciel compilé ou installé, documentation créée, etc.) sans nécessairement refaire toutes les étapes.

`make` est un outil indispensable aux développeurs car :

- `make` assure la compilation séparée automatisée : plus besoin de saisir une série de commandes
- `make` permet de ne recompiler que le code modifié : optimisation du temps et des ressources
- `make` utilise un fichier distinct contenant les règles de fabrication (`Makefile`) : mémorisation de règles spécifiques longues et complexes

- `make` permet d'utiliser des commandes (ce qui permet d'assurer des tâches de nettoyage, d'installation, d'archivage, etc ...) : une seule commande pour différentes tâches
- `make` utilise des variables, des directives, ... : facilite la souplesse, le portage et la réutilisation

De nos jours, les fichiers `Makefile` sont de plus en plus rarement générés à la main par le développeur mais construit à partir d'outils automatiques tels qu' `autoconf`, `cmake`, `qmake`, etc. qui facilitent la génération de `Makefile` complexes et spécifiquement adaptés à l'environnement dans lequel les actions de production sont censées se réaliser.

Liens :

- https://fr.wikipedia.org/wiki/Autoconf
- https://fr.wikipedia.org/wiki/CMake
- http://tvaira.free.fr/dev/cours/cmake.pdf ou https://btssn-lasalle84.github.io/guides-developpement-logiciel/guides-pdf/cmake.pdf
- https://doc.qt.io/qt-6/qmake-manual.html

Voir aussi : Gradle est un moteur de production fonctionnant sur la plateforme Java. Il permet de construire des projets en Java, Scala, Groovy voire C++.

2.4. Le débogueur

Le débogage est un processus de diagnostic, de localisation et d'élimination des erreurs des programmes informatiques. Le débogage permet d'obtenir des programmes qui donnent des résultats corrects.

On utilise un débugueur (de l'anglais *debugger*) ou débogueur (de la francisation bogue).

Un débogueur est un logiciel qui aide un développeur à analyser les *bugs* (bogues) d'un programme. Pour cela, il permet d'exécuter le programme en pas-à-pas, d'afficher la valeur des variables à tout moment, de mettre en place des points d'arrêt sur des conditions ou sur des lignes du programme ...

Le programme à débuguer est exécuté à travers le débogueur et s'exécute normalement. Le débogueur offre alors au programmeur la possibilité d'observer et de contrôler l'exécution du programme.

GNU Debugger, également appelé `gdb`, est le débogueur standard du projet GNU. Il est portable sur de nombreux systèmes type Unix et fonctionne pour plusieurs langages de programmation, comme le C et le C++. Il fut écrit par Richard Stallman en 1988. `gdb` est un logiciel libre, distribué sous la licence GNU GPL. L'interface de `gdb` est une simple ligne de commande, mais il existe des applications qui lui offrent une interface graphique beaucoup plus conviviale.

Pour débuguer un programme avec `gdb`, il faut l'avoir compilé avec l'option `-g` de `gcc/g++`.

Notons également que `gdb` est souvent invoqué en arrière-plan par les environnements de développement intégré (IDE).

Support de cours : gdb et version diapos

Pour le débogage d'application Web, il faut utiliser les **outils de développement** fournis par certains navigateurs comme Chrome DevTools et Firefox DevTools. Et les systèmes de validation du W3C : http://validator.w3.org/

2.5. La documentation

L'accès à la documentation (des langages informatiques, API, bibliothèques, *framework*, ...) est probablement la pierre angulaire du travail d'un développeur.

On peut citer :

- Les pages `man` : `man` est une commande disponible sur les systèmes d'exploitation de type Unix. Elle permet de visionner les contenus d'une documentation. À l'origine, elle sert à accéder aux manuels des commandes d'un *shell* Unix et à la description des fonctions du langage C. Voir aussi : http://tvaira.free.fr/dev/methodologie/creation_pages_man.pdf
- Le service MSDN et **MSDN Library** : C'est la référence incontournable (car officielle) pour tout développement logiciel sur Windows. Elle regroupe l'intégralité des API Windows, du plus bas niveau (écriture de drivers) au plus haut (développement de modules pour les produits Microsoft).
- Les sites web : les éditeurs fournissant généralement un site web pour la documentation, par exemple : JavaScript, jQuery API, PHP, ...

Quelques outils :

Un navigateur (*offline*) de documentation API et un gestionnaire d'extraits de code (*snippet*) :

- Windows et GNU/Linux : Zeal
- macOS : Dash

> Un *snippet* est un terme de programmation informatique désignant une petite portion réutilisable de code source ou de texte. La gestion de snippets est une fonctionnalité de certains éditeurs de texte et des IDE.
>
> Par exemple : GitHub Gist

Un moteur de recherche :

- hello : un moteur de recherche pour développeurs
- searchcode, Krugle, CodeGravity, snipplr, PublicWWW

Un forum pour développeurs :

- Stack Overflow, Developpez.com, ...

3. Environnements de développement intégré

Un environnement de développement intégré (IDE ou EDI en français) est un ensemble d'outils qui permet d'augmenter la productivité des programmeurs qui conçoivent des logiciels. En aucun cas, ils doivent être considérés comme des outils indispensables ou obligatoires.

Un EDI comporte généralement un **éditeur de texte** destiné à la programmation, des fonctions qui permettent, par pression sur un bouton, de démarrer le **compilateur** ou l'**éditeur de liens**, d'**exécuter** le programme ainsi qu'un **débogueur** en ligne, de consulter la **documentation**, etc ...

> Certains environnements sont dédiés à un langage de programmation (IDLE pour Python par exemple), un *framework* (Qt Creator pour Qt par exemple, Android Studio pour Android) ou une architecture (Xcode pour Mac OS X et iOS par exemple) en particulier.

Exemple : Visual Studio Code (VSCode)

VSCode est un « éditeur de code » développé par Microsoft sous licence open source (MIT) et disponible pour Windows (7, 8 et 10), macOS (10.10 et supérieur) et GNU/Linux. Il utilise des extensions pour devenir pleinement un EDI ou IDE.

> Dans le sondage auprès des développeurs réalisé par Stack Overflow en 2021, Visual Studio Code a été classé comme l'outil d'environnement de développement le plus populaire, avec 71,06 % des 82 277 répondants déclarant l'utiliser.

Les fonctionnalités incluent la prise en charge du débogage, la mise en évidence de la syntaxe, la complétion intelligente du code, les *snippets*, la refactorisation du code et Git intégré.

Liens:

- https://code.visualstudio.com/
- https://marketplace.visualstudio.com/vscode
- https://code.visualstudio.com/docs/getstarted/keybindings
- https://code.visualstudio.com/docs/getstarted/tips-and-tricks

Notions à approfondir :

- Espace de travail (*workspace*)
- Paramètres
- Palette de commandes
- Les extensions

Liens :

- Premiers pas avec Visual Studio Code (VSCode) ou vscode.pdf
- Guide Visual Studio Code (VSCode)
- VSCode et Git
- L'extension PlatformIO pour le développement sur systèmes embarqués (Arduino, ESP32, ...)

4. Environnements de développement en ligne

Il existe de nombreux sites web qui fournissent des EDI (Environnement de Développement Intégré) en ligne pour tester du code ou des services : un espace d'apprentissage séparé. Ils permettent aussi d'échanger des exemples.

Quelques sites :

- Coding Ground For Developers : https://www.tutorialspoint.com/codingground.htm pour tout !
- JSFiddle : https://jsfiddle.net/ pour HTML, CSS et JavaScript
- Codeply : https://www.codeply.com/ pour les frameworks JavaScript
- C++ Shell : https://cpp.sh/
- ...
- Visual Studio Code Online : https://vscode.dev/
- Gitpod : https://www.gitpod.io/
- Codeanywhere (Cloud IDE) : https://codeanywhere.com/
- ...
- Wokwi (Simulateur en ligne Arduino et ESP32) : https://wokwi.com/

En informatique, le bac à sable (*sandbox*) est une zone d'essai permettant d'exécuter des programmes en phase de test ou dans lesquels la confiance est incertaine. C'est notamment très utilisé en sécurité informatique pour sa notion d'isolation.

5. Autres outils

5.1. Rédaction de documents

Cet aspect est déjà abordé dans ce guide HTML (PDF).

5.2. Gestion de versions

Un logiciel de gestion de versions (VCS pour *version control system*) est un logiciel qui permet de stocker un ensemble de fichiers en conservant l'historique de toutes les modifications effectuées, y compris par plusieurs personnes.

Il permet de retrouver le code source dans l'état où il était à une date donnée, à une version donnée.

Les fichiers versionnés sont mis à dispositions sur un dépôt (*repository*), qui est un espace de stockage géré par un logiciel de gestion de versions.

Les différentes versions (ou révision) sont nécessairement liées à travers des modifications : une modification est un ensemble d'ajouts, de modifications, et de suppressions de données.

Essentiellement utilisée dans le développement logiciel, elle concerne surtout la gestion des codes source.

On distingue :

- la gestion de versions **centralisée** (comme CVS et Subversion (SVN)) : il n'existe qu'un seul dépôt (*repository*) des versions qui fait référence.
- la gestion de versions **décentralisée** (comme Git et Mercurial) : il existe plusieurs dépôts pour un même logiciel. Un DVCS offre les mêmes services qu'un VCS sur une architecture décentralisée (ou distribuée).

L'auteur de développement logiciel Joel Spolsky, ancien chef de projet de Microsoft Excel et co-créateur du site Stack Overflow, décrit la gestion de version décentralisée comme « probablement la plus grande avancée dans les technologies de développement logiciel dans les 10 [dernières] années. ».

Depuis les années 2010, Git est le logiciel de gestion de versions le plus populaire dans le développement logiciel et web, qui est utilisé par des dizaines de millions de personnes, sur tous les

environnements (Windows, Mac, Linux). Git est aussi le système à la base du célèbre site web GitHub, le plus important hébergeur de code informatique.

Liens :

- Guide Git : HTML | PDF
- Support de cours Git : PDF

Il est possible d'héberger des dépôts Git sur un site externe dédié à l'hébergement : GitHub, GitLab, Bitbucket, ...

5.3. Documentation automatique de code

Le développeur aura nécessairement besoin de générer une documentation du logiciel.

Doxygen est un système de documentation pour C, C++, Java, Python, Php et autres langages. Il permet de générer la documentation des développements :

- à partir des commentaires insérés dans le code source
- à partir de la structure du code lui même

La documentation peut être produite dans des formats variés tels que du HTML, du Latex, du RTF ou du XML.

Guide : HTML | PDF

5.4. Tests logiciels

Le test est une recherche d'anomalie (défaut, appelé souvent *bug*) dans le comportement d'un logiciel.

Si les tests se déroulent parmi les activités finales du projet, ils se préparent dès le début. Il existe différentes formes de tests :

- Tests de fonctionnements (unitaires et intégrations) : pour vérifier que le produit est bien fait
- Tests de validation : permettent de vérifier que le produit réalisé est le bon
- Tests d'architecture : pour vérifier que le produit est utilisable dans son environnement d'exploitation

En détaillant ces tests dans l'ordre dans lequel ils sont effectués :

- Tests unitaires : ils sont planifiés lors de la conception détaillée. Ils permettent de tester "les plus petites unités testables": méthodes, fonctions, etc.
- Tests d'intégrations : planifiés en analyse et en conception préliminaire, ils évaluent les différentes unités intégrées, packages, groupement de classes, mais aussi les interactions matérielles - logiciel en cas de système embarqué.
- Tests de validations : conçus dès les spécifications, ils permettent de vérifier l'adéquation du

produit aux exigences fonctionnelles du client.

Supports de cours :

- Tests logiciels
- Autres tests
- Méthodes de tests
- Tests unitaires
- Tests de validation

Quelques outils de tests unitaires :

- CppUnit
- CxxTest
- PHPUnit

5.5. Analyse de code source

Objectif : diagnostiquer et corriger les erreurs de programmation typiques, telles que les violations de style, …

> `lint` est une commande UNIX de préprocesseur permettant l'analyse statique de code source en langage C. L'utilitaire `lint` a été l'un des premiers outils d'analyse statique de code source.

Quelques outils :

- `lint` : analyse statique de code source C
- `clang-tidy` : « linter » C++ basé sur `clang`
- `cppcheck` : analyse de code statique pour les langages C et C++
- `gprof` : effectue du profilage de code
- `gcov` : analyse de la couverture du code source
- `valgrind` : mettre en évidence des fuites mémoires

Exemples d'utilisation de clang-tidy et cppcheck

5.6. Installateurs

Exemples : Inno Setup et Qt-Installer-Framework

Les paquetages sous GNU/Linux :

- Paquet DEB
- Paquet RPM

Voir aussi : Script autoextractible

5.7. Gestion de projets

Pour développer un logiciel, un système de gestion de projets sera indispensable.

Quelques outils :

- À l'origine, Jira est un système de suivi de bugs et de gestion des incidents (tickets). Il est maintenant un système de gestion de projets développé par Atlassian.
- Trello est un outil de gestion de projet en ligne, inspiré par la méthode Kanban de Toyota.

Guides :

- Jira (PDF) et le lien avec GitHub, GitLab, Bitbucket : HTML | PDF
- Trello et Beesbusy
- Gestion de projets

5.8. Diagrammes

Les outils de dessins orientés sur les diagrammes de logiciel utilisent souvent la notation UML (*Unified Modeling Language*).

Il existe de nombreux logiciels de modélisation UML. On peut citer BOUML.

Lien : Comparaison des logiciels d'UML

Les autres diagrammes seront réalisés avec tout outil de dessin assisté par ordinateur (DAO). On peut citer diagrams.net (auparabant **draw.io**). Il existe sous forme d'extension pour VSCode.

Voir aussi :

- Mermaid
- PlantUML
- Graphviz et http://magjac.com/graphviz-visual-editor/

5.9. Prototypage IHM

Il existe de très nombreux logiciels permettant de *designer* des interfaces utilisateurs, par exemple Balsamiq.

Quelques liens :

- https://www.figma.com/
- https://pidoco.com/
- https://www.invisionapp.com/inside-design/design-resources/

- https://www.diagrams.net/
- https://precursorapp.com/home

5.10. Atelier de génie logiciel

Un atelier de génie logiciel (AGL) permet la construction, la conception et la planification des travaux. Il comporte tous les outils présents dans un environnement de développement intégré, plus des outils de conception, de planification, de test, et des outils créant automatiquement du code source et de la documentation.

5.11. Virtualisation et Conteneurs

La virtualisation consiste à exécuter sur une machine hôte dans un environnement isolé :

- des systèmes d'exploitation : virtualisation système
- des applications : virtualisation applicative

Support de cours : http://tvaira.free.fr/bts-sn/admin/Virtualisation.pdf

> Voir aussi : http://tvaira.free.fr/bts-sn/admin/Presentation-Vagrant.pdf et http://tvaira.free.fr/bts-sn/admin/Creation-Box-Vagrant.pdf

Docker est une plateforme permettant de lancer certaines applications dans des conteneurs logiciels. Il s'agit actuellement du moteur de conteneurisation le plus utilisé.

Support de cours : Docker

Devox est un outil en ligne de commande qui permet de créer facilement des *shells* et des conteneurs isolés. Devbox crée un environnement isolé uniquement pour l'application à partir d'une liste des *packages* requis par l'environnement de développement.

5.12. Boîte à outils

Quand on est développeur, il y a parfois certains utilitaires dont on a besoin qui nécessite de chercher un site web ou un petit outil pour le faire rapidement. Cela peut-être générer un texte en *lorem ipsum*, formater un JSON, décoder un Base64, tester une regex et …

Tout cela peut être regroupé dans une boîte à outils, comme DevBox (payant) ou version en ligne.

Sous Windows : DevToys

Sous macOs : DevUtils et DevToysMac

6. Voir aussi

- Chaîne d'outils Devops