LINUX / IA / DENO / BASIC / BLAZOR / ESP / KOTLIN

PHP / WINDOWS / SQL / THREADS / C++ / ROOT ME / LARAVEL / GIT

# PROGRAMMEZ!

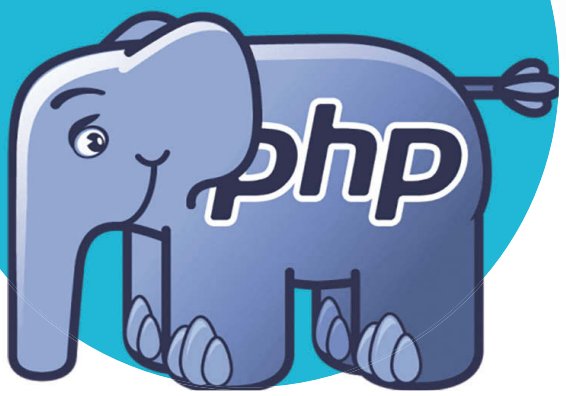
Le magazine des développeurs

02/2020 N°237 22e année

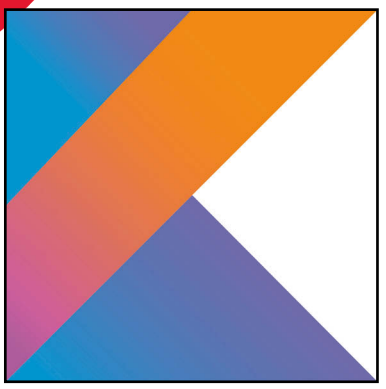
# DENO

## Le nouveau back-end JavaScript

## Tout savoir sur PHP7.4




## Le retour de DELPHI sur LINUX DX avec FMX LINUX



# Kotlin

## pour les dévs Java

# Blazor :

## une techno prometteuse

M 04319 - 237 - F: 6,50 € - RD

Le seul magazine écrit par et pour les développeurs

Printed in EU - Imprimé en UE - BELGIQUE 7 € - Canada 9,80 $ CAN - SUISSE 13,10 FS - DOM Surf 7,50 € - TOM 1020 XPF - MAROC 55 DH

WALLIX

CYBERSECURITY SIMPLIFIED

# PRENEZ DE LA HAUTEUR

## EN GARDANT L'ESPRIT D'ÉQUIPE !

### REJOIGNEZ UN LEADER EUROPÉEN DE LA CYBERSÉCURITÉ

A l'ère de la transformation numérique, les enjeux de cybersécurité touchent l'ensemble des entreprises. WALLIX a réussi à combiner trois exigences fondamentales : la gestion des identités, le contrôle des accès et la protection des données, grâce à des solutions simples à utiliser et faciles à déployer. Les solutions WALLIX accompagnent aujourd'hui plus de 1000 entreprises et organisations dans la gestion de leur cybersécurité, la sécurité des données et la conformité réglementaire.

PRENEZ DE LA HAUTEUR ! REJOIGNEZ UN LEADER !

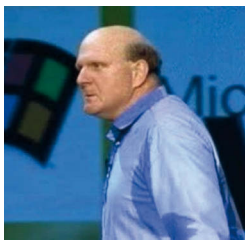
La culture WALLIX, fortement imprégnée d'un esprit entrepreneurial, passe par des valeurs humaines comme l'audace, l'engagement, un fort esprit d'équipe, et le goût du défi... ET C'EST TOUT VOUS !

Plus d'informations sur www.wallix.com

# « Developers Developers Developers »

La motivation te gagne ? Tu veux la montrer ? Il existe une solution.

Elle est simple : le niveau Steve Ballmer dans son meeting de motivation. Mais là c'est le niveau ultime de la motivation. Il faut sauter partout, être motivé comme jamais, une salle chauffée à 30° bien humide et une bonne sueur pour montrer que ta motivation est à la hauteur du défi à relever… afficher un hello world en 5 lignes de code… en Java… (là faut être motivé non ?)



Toutes et tous ensemble, crions (idéalement sans respirer sinon c'est pas drôle) :
Developers Developers Developers Developers Developers
Developers Developers Developers Developers Developers
Developers Developers Developers Developers Developers
Developers Developers Developers Developers Developers

Le refrain peut être repris dans le désordre. Si vous avez la version elfique ou klingeon, c'est encore mieux mais légèrement moins compréhensible par 99,95 % de la population.

La version Pokémon est aussi une option.


Avertissement : le Steve Ballmer de l'an 2000 peut générer une Race Condition. Donc méfiance…

**François Tonic** - ftonic@programmez.com

# SOMMAIRE

Brèves ——— 4
Agenda ——— 6
Roadmap ——— 8
Carrière ——— 9
Matériel ——— 12
Aoz Studio ——— 14

**Abonnez-vous !** ——— 42
**Commitstrip** ——— 82

SÉCURITÉ

Blackhat .....17
YesWeHack Edu....18

Root Me ..19

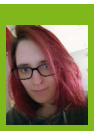
DOSSIERS


Delphi revient sur Linux .....24

Deno : le nouveau serveur TS.....29

Blazor : la nouvelle techno tendance ? .....33

NIVEAU 100


Lavarel .....41

SystemD .....50
Kotlin .....51

PHP 7.4 .....47

Git *partie 2* .....54

NIVEAU 200


ESP + IoT *partie 1* .....57

SQL avancé *partie 2* .....67

Développement Linux sur Windows 10 .....60

Venom *partie 2* .....69

Architecture hexagonale *partie 2* ....64

Contour *partie 2* .....70

NIVEAU 300

Thread en Java *partie 1* .....75

C++ & menu Windows .....79

**MESSAGE PRIORITAIRE :**

**Programmez! n°238, prochain numéro date stellaire : 28.02.2020**

*DevOps côté dévs et les nouvelles tendances du DevOps*
*Go : la nouvelle star du développement*
*Informatique quantique*

## Cedric O veut aller vers la reconnaissance faciale

Le secrétaire d'État au numérique souhaiterait mettre en place une période d'essai pour un dispositif de vidéosurveillance boosté à la reconnaissance faciale en 2020. L'idée a été lancée à l'occasion d'une interview sur le sujet d'Alicem, le projet d'authentification par reconnaissance faciale porté par le gouvernement. Preuve que cette technologie ne se limitera pas à la simple authentification, mais que le gouvernement aimerait bien l'utiliser pour la surveillance. Simple petit problème évoqué par Cedric O : le RGPD interdit purement et simplement ce type d'usage. Reste à savoir comment contourner cet obstacle à l'innovation.

## Internet : La Russie largue les amarres

Après l'avoir annoncé, la Russie a finalement mené à bien ses tests visant à déconnecter son infrastructure Internet du reste du réseau mondial. L'objectif pour la Russie est d'assurer sa capacité à déconnecter complètement le cyberespace russe du reste du monde afin « de protéger les citoyens en cas d'attaque. » Le gouvernement russe n'a donné aucun détail technique sur la nature de ces tests et s'est contenté d'affirmer que ceux-ci avaient été concluants : on prendra donc les pincettes qui s'imposent, la Russie étant connue pour ses effets de manche.

## Microsoft : après la Russie, la Corée du Nord

Microsoft a annoncé avoir fait suspendre 50 noms de domaines utilisés selon eux par le groupe Thallium, lié au gouvernement de la Corée du Nord. Selon Microsoft, ces noms de domaines imitaient des propriétés intellectuelles de la firme de Redmond afin de mener à bien des attaques de phishing contre des cibles diverses allant d'universités à des personnes travaillant sur la dissuasion nucléaire. Microsoft dispose en son sein d'un cabinet de juristes entièrement dédiés à chasser les utilisations frauduleuses de ses marques par des attaquants, mais jusqu'alors ils se concentraient principalement sur des groupes de cybercriminels russes.

## L'armée américaine resserre la vis sur le « cyber »

Alors que les tensions avec l'Iran reprennent, l'armée américaine semble vouloir se prémunir des attaques informatiques. Le ministère américain des armées a ainsi passé plusieurs directives concernant les soldats américains : les soldats envoyés en mission au Moyen-Orient ont récemment reçu l'ordre de laisser leurs appareils électroniques chez eux. Un peu plus tôt, le gouvernement avait purement et simplement interdit d'utilisation l'application TikTok, considérée comme un potentiel risque de sécurité. Et le ministère américain de l'Intérieur a également fait passer une circulaire mettant en garde sur les capacités cyberoffensives de l'Iran. Autant dire qu'outre-Atlantique, la cyberguerre n'est plus un vain mot.

## Magasins automatisés : Casino se heurte à ses syndicats

Casino a été contraint de reporter une expérimentation visant à ouvrir une centaine de ses magasins le dimanche de façon autonome, en n'utilisant que des caisses automatisées. Une première expérimentation avait eu lieu le 15 décembre et plusieurs employés étaient venus manifester à l'appel de la CGT. Suite à ce premier épisode, c'est une union intersyndicale (CGT, FO, CFDT) qui s'est levée pour dénoncer cette expérimentation, qui vise tout simplement à se passer des hôtesses de caisse le dimanche au profit des bornes automatisées.

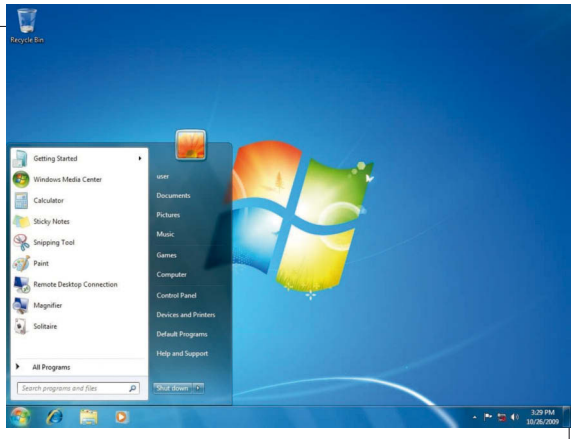
## Amazon déploie peu à peu son concept Amazon Go, des magasins « automatiques »


© SounderBruce

## Patch Tuesday : la NSA s'en mêle

Le premier patch tuesday de Microsoft pour l'année 2020 était regardé de très près et pour cause : la NSA donnait à cette occasion une conférence de presse afin d'alerter sur une faille de sécurité découverte par ses soins au sein des mécanismes de chiffrement de Windows. L'agence indique que celle-ci peut être utilisée pour manipuler notamment les mécanismes de signature numérique des logiciels dans Windows 10 et sur Windows Server 2016 et conseille aux utilisateurs de patcher. Quand la NSA se donne la peine de le faire, cela mérite d'être pris en compte.

## Windows 7 : cette fois c'est la bonne ?



On le savait depuis un moment, mais la date fatidique est enfin arrivée : le 14 janvier 2020, le support de Windows 7 cesse purement et simplement. Le système d'exploitation de Microsoft ne sera donc plus mis à jour et les éventuelles vulnérabilités découvertes sur le logiciel ne seront donc pas corrigées. Sauf évidemment pour ceux qui sont prêts à mettre la main à la poche et à payer pour avoir droit aux Extended Security Update, des mises à jour de sécurité qui seront proposées jusqu'en 2023. Fin 2019, Netmarketshare estimait ainsi la part d'appareils fonctionnant encore sous Windows 7 à un peu plus de 32 %, soit un marché non négligeable pour ceux qui rechignent encore à upgrader.

25 février
MEETUP#8 PROGRAMMEZ
# Focus sur Flutter

Mars
MEETUP#9 PROGRAMMEZ
# Un monde sans OS ?

**par François Tonic (Programmez!)**

**A PARTIR DE 18H30.**

Où : Infeeny 5 rue d'Uzès Paris
Métro : station Grands Boulevards (lignes 8 & 9)

**Informations & inscription : programmez.com**

## Février

### 1 & 2 : FOSDEM 2020/Bruxelles

L'évènement incontournable des développeurs open source. C'est l'occasion de découvrir des projets et surtout d'autres développeurs. On échange, on discute, on oppose les arguments.
Site : https://fosdem.org/2019/

### 14 : DevFest Paris/Paris

La DevFest revient à Paris mi-février. L'évènement promet de superbes sessions, des animations, et beaucoup de technos et de rencontres ! 36 sessions sont prévues. Attention : l'évènement change de lieu. Il s'installe au palais des congrès d'Issy-les-Moulineaux.
Site : https://devfest.gdgparis.com

### 28 : DevFest du bout du monde/Brest

C'est à la pointe de la Bretagne. Et on va y parler langages, codes et technologies. L'évènement breton pour les développeurs !
site : https://devfest.duboutdumonde.bzh

## Mars

### 17 : AWS Summit Paris/Paris

Amazon Web Services propose sa grande journée française pour présenter les services, les supports et les partenaires. De nombreuses sessions et des keynotes !
Site : https://aws.amazon.com/fr/events/summits/paris/

### 25-27 : BreizhCamp/Rennes

Le BreizhCamp, c'est 3 jours de conférence à Rennes. Créé en 2011 à l'initiative du BreizhJUG, nous sommes en cours de planification de la 10e édition. Le BreizhCamp propose de se faire rencontrer une communauté de développeurs et d'experts, avec un contenu à la carte sur plus de 100 thèmes présentés. Chaque participant est libre de suivre les sujets qui ont retenu son attention ou de préférer les ateliers pour mettre en pratique les connaissances acquises.
https://www.breizhcamp.org

## Avril

### 15-17 : Devoxx France/Paris

Devoxx France c'est du 15 au 17 avril 2020. La 9e édition s'annonce très bien. La première nouvelle c'est qu'il n'y a plus de places à vendre... depuis l'ouverture. Jeudi et vendredi matin, il y aura 5 à 6 plénières, sur un format de 20mn, proche de ce que font les conférences TED. Le thème de cette année est "Tech 4 good, Tech 4 evil". Comment notre métier et la technologie peuvent-ils améliorer (ou non) la vie au quotidien des clients ? Une personne de la Fondation de l'Abbé Pierre, un chercheur, une CEO dans l'humanitaire... Sans dévoiler le programme organisé par Benoit Lafontaine, Katia Aresti et Charlotte Abdelnour, on peut s'attendre à belles choses ! +920 sujets reçus pour 240 sessions ! Bref, c'est l'évènement incontournable des dévs.

### 24 : Serverless Days / Paris

L'architecture serverless se répand rapidement dans les entreprises et les infrastructures. Une journée complète pour discuter avec les meilleurs experts du domaine : https://paris.serverlessdays.io

### 29 & 30 : MixIT/Lyon

De nombreux thèmes sont abordés : design, technologie, makers, éthique dans l'IT, style de vie, travail en équipe, etc. https://mixitconf.org

## Mai

### 6-7 : GitHub Satellite/Paris

La conférence européenne de GitHub s'installe pour la première fois en France. Durant 2 jours, l'écosystème GitHub fait le show et l'éditeur propose des dizaines de sessions et d'ateliers. Un évènement à ne pas rater. Site : https://githubsatellite.com

### 13-15 : RivieraDev/Sophia Antipolis

La conférence développeur du sud revient. 42 conférences, 18 ateliers et +500 développeurs présents. Le 13 est réservé aux sessions intensives de 3h avec des experts niveau 300 ! Les 14 et 15 sont les journées classiques des sessions.

**Merci à Aurélie Vache pour la liste 2020, consultable sur son GitHub :**
https://github.com/scraly/developers-conferences-agenda/blob/master/README.md

## Juin

### 4-5 : Best of web 2020/Paris

Best of web, c'est un évènement sur deux jours. Une journée de workshops et une journée de conférence préparée par 16 meetups web parisiens. La journée-conférence est composée à 50 % du Best Of des talks meetups de l'année et 50 % d'inédits provenant d'une CFP. Les tickets sont spécifiques à chaque journée.
Site : http://bestofweb.paris

### 12/DevFest Lille/Lille

La DevFest revient à Lille. L'appel aux speakers est lancé. C'est le plus grand évènement dans le nord, avec +800 personnes attendues.
Site : https://devfest.gdglille.org

## Octobre

### 30 : Agile Tour Brussels/Bruxelles

C'est l'évènement agilité en Belgique. +200 personnes attendues.
Site : http://www.agiletourbrussels.be

## LES CONFÉRENCES DOT EN FRANCE

### 3 février : dotSwift / Paris

La conférence Swift revient à Paris. Avec les dernières évolutions du langage, le nouveau framework UI, il y aura de nombreuses choses à découvrir.

### 2 mars : dotPy/Paris

Une des plus importantes conférences en France sur le langage Python !
https://www.dotpy.io

### 30 mars : dotGo/Paris

Comme son nom l'indique, la conférence dot 100 % langage Go !

NOUVEAU

# Retour vers le passé :

*redécouvrez les ordinosaures et les technologies des années 1970 à 2000 !*


Commandez directement sur **programmez.com**

**6,66 €** *(+frais de port*)* **36 pages**

*Revue trimestrielle. Editée par Nefer-IT. *Avec frais de port : 7,66 €*

# Roadmap des langages

*Chaque mois, Programmez ! vous propose un panorama des agendas de sorties des versions des langages, frameworks, etc.*

## Flutter 1.12

Cette version propose une belle panoplie de nouveautés et améliorations. Citons :

- Le support du mode sombre iOS 13
- Version alpha de Flutter pour macOS
- Flutter pour le web arrive en bêta
- La version de DART évolue aussi. Flutter supporte la v2.7
- Meilleure stabilité des API pour Java, Kotlin, Objective-C, Swift

Cette version est sortie en décembre. Depuis, de nombreuses mises à jour ont été déployées. La dernière en date est la 1.12.13 : https://flutter.dev/docs/development/tools/sdk/release-notes/release-notes-1.12.13#breaking-changes

## DART 2.7

Les deux grosses nouveautés de Dart 2.7 sont la prise en charge des méthodes d'extension, ainsi qu'un nouveau package pour la gestion des chaînes avec des caractères spéciaux.
Les méthodes d'extension permettent d'ajouter de nouvelles fonctionnalités à n'importe quel type, même les types que vous ne contrôlez pas.

## Kotlin 1.3.70

Cette future version annonce de nouvelles choses :

- Meilleur support de gradle.kts. Les développeurs veulent améliorer le support des fichiers et réduire la charge CPU durant la synchronisation
- Sur Kotlin/JavaScript : on attend une optimisation sur les navigateurs. Des tâches vont changer de nom comme browserWebPack. Donc méfiance.

Lien : https://discuss.kotlinlang.org/t/kotlin-1-3-70-early-access-preview/15876

## TypeScript 3.8

Actuellement, la 3.8 est en bêta. Cette mouture se fait principalement remarquer par une nouvelle fonctionnalité permettant d'importer/exporter des types seuls. On notera aussi le support de l'ECMAScript Private Fields.
Lien : https://devblogs.microsoft.com/typescript/announcing-typescript-3-8-beta/

## Angular 10.0

Version attendue pour mai 2020

## Python 3.8.1

La version majeure 3.8 est sortie en automne dernier. Actuellement, nous sommes à la v3.8.1. Il s'agit d'une version de maintenant.

## Symfony

Le framework PHP évolue tranquillement, mais sûrement. Actuellement, on peut utiliser la version 4.4 et 5.0.

**Côté support :**
4.4.x : sortie en novembre 2019, support actif jusqu'au novembre 2022
5.0.x : support jusqu'en juillet 2020
5.1 : sortie prévue en mai et fin support janvier 2021

Parmi les nouveautés disponibles, on notera les nouvelles méthodes Dom Crawler et la gestion des secrets cryptés.

## PHP 7.4

Disponible depuis fin novembre 2019. Cette version apporte de nombreuses évolutions et nouveautés : voir article dans ce numéro.

## Ruby 2.7

Disponible depuis fin 2019. On y remarque principalement l'introduction du filtrage de motif à titre expérimental. Le filtrage de motif, ou pattern matching, est une fonctionnalité très répandue en programmation fonctionnelle, mais plus rare dans les langages impératifs et/ou orientés objet. Le filtrage de motif est très apprécié en programmation fonctionnelle, car il apporte de la clarté et de la concision au code.
Cette version propose aussi la capacité du ramasse-miettes à défragmenter de la mémoire fragmentée et ainsi limiter l'empreinte mémoire des applications ou la dégradation de leurs performances.

## Java 14

***Date de sortie :*** mars
***Les principales nouveautés attendues/annoncées :*** Java 14 proposera assez peu de choses à en croire les premières builds. On notera le JFR Event Streaming (consommation des données JFR). Il permettra de collecter des données de profilage et d'analyse d'une application Java en exécution. On aura aussi les expressions switch. Elles doivent simplifier le codage. NullPointerException aura droit à quelques améliorations. Java supporta aussi les mémoires NVM.

## C++20

Courant 2020. Les spécifications de C++20 sont désormais figées ; un premier document complet sera disponible cet été. Cette future version du langage mettra en avant deux nouveautés : les modules et les coroutines. Les modules constituent une nouvelle alternative aux fichiers d'en-tête qui apportent un certain nombre d'améliorations clés, notamment en isolant les effets des macros et en permettant des compilations évolutives, explique Herb. Il ajoute qu'en 35 ans c'est la première fois que C ++ ajoute une nouvelle fonctionnalité permettant aux utilisateurs de définir une limite d'encapsulation nommée. Les coroutines sont elles aussi une fonctionnalité à remarquer. Une coroutine est une unité de traitement qui s'apparente à une fonction (ou routine), avec cette différence que si une sortie du corps d'une fonction met fin à l'exécution de celle-ci, la sortie de la coroutine suspend seulement son traitement qui peut ensuite reprendre, l'état du traitement à la sortie étant conservé. Une coroutine peut être vue comme un morceau de programme qui conserve son état, mais qui n'a pas de thread d'exécution. Les coroutines peuvent notamment être utilisées pour des itérateurs et des générateurs.

## .Net 5

Novembre 2020. L'annonce en a été faite à la dernière conférence BUILD. Il s'agit de .Net Core vNext, donc au-delà de .Net Core 3.0. Il s'agit de réunifier les noms. L'ambition est d'être disponible sur Windows, Linux, macOS, iOS, Android, tvOS, webassembly, etc.

**Aurélie Vache**
*Cloud Dev(Ops) chez Continental*
*Duchess France & DevFest Toulouse Leader*
*Toulouse Data Science core-team &*
*Marraine Elles bougent*
@aurelievache

# La reconversion, c'est maintenant !

Partie 2

Pourtant, on entend encore des personnes qui font l'amalgame entre le métier de Développeur et le fait d'être obligatoirement un ingénieur. Il y a une différence entre sa formation et le métier.

"Développeur" ! = "Ingénieur".

Ces dernières années ont vu l'arrivée de formations courtes pour devenir développeur : O'clock, Simplon, Wild code school…

Est ce que ces formations valent vraiment le coup ? Devient-on développeur en sortant ? Les formations prodigués par les différents organismes de formation sont-elles équivalentes ? Nous verrons plusieurs organismes de formation spécialisés dans la reconversion pour être développeur.

## Les formations

Trouver un organisme de formation et une formation pour se reconvertir en développeur, répondant à ses valeurs et à ses besoins, peut prendre beaucoup de temps. Afin de vous faciliter la vie, vous trouverez ci-dessous un topo sur quelques organismes de formation, et j'espère que cela vous aidera si vous ou vos proches souhaitent se reconvertir.

## SIMPLON

Simplon est une entreprise sociale et solidaire qui propose des formations gratuites et intensives aux métiers techniques du numérique, en s'adressant prioritairement à des profils peu représentés dans ce secteur : demandeur·se·s d'emploi, non diplômé·e·s, issu.e.s des quartiers prioritaires, zones rurales, avec un objectif de parité femmes-hommes. Simplon est labellisée French-Tech et Grande École du Numérique.

Chez Simplon, la démarche pédagogique repose sur plusieurs piliers :

- **UNE PÉDAGOGIE ACTIVE** qui favorise l'apprentissage en faisant, le mode projet, le co-apprentissage, pair à pair, etc.
- **UNE ÉQUIPE PLURIDISCIPLINAIRE** : un formateur référent, des experts techniques, des coordinateurs pédagogiques et des équipes projet.
- **L'ENTREPRISE AU COEUR DU PARCOURS** : des actions de suivi avec les équipes RH et les tuteurs en entreprise, des professionnels qui viennent témoigner de leur parcours, de la réalité du terrain et dispenser des master class, des projets pédagogiques émanant de besoins concrets d'entreprises.

Avec 75% de sortie positive 6 mois après la formation (emploi, création d'entreprise et poursuite de formation) Simplon accompagne aussi chaque apprenant dans son projet et son insertion professionnelle, pendant et après la formation. En Occitanie, Simplon déploie plusieurs programmes de formation destinés aux demandeur/ses d'emploi.

### Des actions d'acculturation aux métiers du numérique

**Le SAS #Hackeuses** : un programme de 6 semaines réservé aux femmes pour s'initier au code, découvrir les métiers techniques du numérique et pourquoi pas valider un projet dans ce domaine. En mai-juin 2019, une vingtaine de femmes ont bénéficié de ce programme à Labège. Une seconde session est prévue sur Montpellier.

**Le SAS #Refugeek** : un programme destiné aux réfugié·es pour acquérir les compétences numériques fondamentales et améliorer son niveau de français.

### Des formations métiers

**Développeur/se web / web mobile et Technicien/ne Supérieur/e Systèmes et Réseaux** : deux parcours de formation de 11 mois dont 3 en entreprise. A la clé : un Titre Professionnel de niveau bac+2

### Des spécialisations dans les métiers du numérique

**Technicien/ne Data** : un nouveau programme de formation certifiante de 7 mois en présentiel et un contrat en alternance d'un an.

**Pour les développeur/se/s, une spécialisation en objet connecté** de 560 heures de formation en présentiel et à distance.

Enfin, Simplon.co propose aussi des formations courtes pour les salarié·es et les demandeur·ses d'emploi sur des thématiques liées au numérique : méthodes agiles, savoir utiliser Wordpress, etc.

**Pour en savoir plus :**
https://simplon.co/occitanie/
Twitter, Facebook, LinkedIn : Simplon Occitanie

## WILD CODE SCHOOL

La Wild Code School (wildcodeschool.fr) est une école innovante et un réseau européen de campus qui forment aux métiers tech des spécialistes adaptables.

***La pédagogie*** hybride repose sur la réalisation de projets, une plateforme pédagogique originale, un accompagnement personnalisé vers l'emploi et une autonomie renforcée des élèves dans l'acquisition des savoirs.

***Les campus*** sont à taille humaine et intégrés dans les écosystèmes numériques locaux au plus près des recruteurs.

***La communauté*** soudée d'élèves et d'anciens élèves est internationale et apporte son soutien à chaque Wilder sur le plan personnel et professionnel. Pour en savoir plus :
https://www.wildcodeschool.com/fr-FR

## O'CLOCK

O'clock est une école de développement web un peu spéciale qui repose sur le format téléprésentiel. Un mot un peu barbare mais qui est en réalité très simple : de l'éducation accessible à distance. L'école a développé sa propre technologie de salles de classe virtuelles, permettant d'accueillir un groupe d'une trentaine d'étudiants qui vont suivre en direct, ensemble et en même temps, des cours quotidiens animés par un développeur-formateur expérimenté. L'idée est donc de remettre le formateur au centre de la transmission du savoir tout en permettant à toutes et à tous de pouvoir accéder à un enseignement de qualité, sans nécessairement habiter dans une grande ville ou sacrifier le présentiel au profit des MOOCs où le taux d'abandon y est largement plus élevé.

L'école propose différentes formations destinées aux débutants, portant essentiellement sur le métier de développeur web et l'apprentissage des technologies les plus recherchées par les recruteurs.

Pour entrer en formation, pas de pré-requis de diplôme, d'âge ou de profil ! Côté financement, les étudiants peuvent solliciter la grande majorité des dispositifs existants comme l'AIF de Pôle Emploi, le Fongecif, le CPF, les financements régionaux et autres leviers plus spécifiques.

Pour en savoir plus : https://oclock.io
Twitter : https://twitter.com/Oclock_io

# La Fintech Lemon Way, au cœur de l'innovation, recrute de nouveaux talents

*L'année 2020 s'annonce comme celle de la croissance pour Lemon Way, qui va créer 30 à 40 nouveaux postes lors des 12 prochains mois. La phase de recrutement est lancée. Vous avez de bonnes connaissances en C#, .Net, Vue.js ? C'est le moment de postuler !*

Le monde de la Fintech est en plein essor. Sa raison d'être ? Développer de nouvelles technologies pour créer des services financiers utiles répondant aux besoins des entreprises et des particuliers. Or, qui dit Fintech dit bien sûr finance, mais surtout technologie, ce qui implique de recruter des programmeurs talentueux.

Tel est le cas de Lemon Way, l'une des pépites de la Fintech française. Son rôle ? Traiter les flux de paiement qui circulent sur les marketplaces B2B, B2C et C2C et les plateformes de financement participatif. Créée en 2007, l'entreprise a vu son activité bondir au cours des dernières années en proposant à de nombreux clients une solution de collecte, de cantonnement, mais également de contrôle des flux financiers. En 2019, ce sont près de 3 milliards d'euros de flux de paiement qui ont ainsi été traités par Lemon Way. Ce travail, méconnu du grand public, est lié aux obligations réglementaires encadrant ce type d'activité. Le fort développement de l'entreprise s'accompagne d'une phase active de recrutement.

## L'objectif : doubler voire tripler les effectifs de l'équipe Tech

« *Demain, afin d'accompagner la croissance de Lemon Way et d'être à la pointe de l'innovation en termes de paiement, nous avons pour objectif de doubler voire tripler les effectifs de notre équipe Tech* » explique Carmen Ionita, Directrice des ressources humaines de Lemon Way. L'entreprise prévoit ainsi de créer 30 à 40 nouveaux postes cette année, dont la plupart au sein de ses équipes de programmation.


Il faut dire que les beaux projets ne manquent pas chez Lemon Way. L'entreprise compte développer cette année de nombreuses fonctionnalités supplémentaires pour améliorer son offre de service. Paiement en plusieurs fois, utilisation de la Blockchain, reconnaissance faciale, initiation de paiement instantané : la liste des projets 2020 est longue !

L'entreprise se trouve au cœur de l'innovation technologique et financière, d'où ce foisonnement de nouveaux projets. « *Chez Lemon Way, notre cœur de métier est celui des nouveaux moyens de paiement. Il s'agit d'un domaine où l'innovation est omniprésente. Cette innovation vient également des évolutions réglementaires qui encadrent nos métiers, qui nécessitent une adaptation permanente* » souligne Mohamed Otmani, VP Engineering de Lemon Way.

## Le double défi : satisfaire les clients mais également le régulateur

En quoi constituent exactement les tâches d'un programmeur chez Lemon Way ? Gilles Réant, Chief Product Officer, explique : « *Notre équipe de développeurs a pour mission principale de concevoir les nouveaux modules que nous ajoutons régulièrement à notre service de paiement. Cette conception se fait en partenariat avec l'équipe produit qui recueille les besoins client et les transforme en tâches à réaliser par l'équipe de développement. Celle-ci intervient de temps à autres pour corriger des anomalies remontées par nos clients* ».

Satisfaire les requêtes des clients reste en effet la mission première des équipes techniques. « *Au cœur de nos préoccupations résident nos clients. Les développeurs que nous recherchons doivent avoir une bonne compréhension des problématiques liées à l'expérience utilisateur. Leur mission : trouver la meilleure solution pour répondre à leurs besoins* » résume Mohamed Otmani.

Le lien avec les clients passe également, de manière très concrète, par la mise à disposition d'une API qui sert d'interface entre les systèmes IT des clients et ceux de Lemon Way. Ici, l'enjeu est double : l'API doit non seulement satisfaire les clients, mais également les régulateurs. « *En 2019, nous avons reconçu notre API à partir de zéro* » explique Gilles Réant. « *Cela nous a permis de satisfaire plusieurs demandes de clients tout en faisant en sorte de respecter les principes RESTful HATEOAS en étant conformes aux exigences de la seconde directive européenne sur les services de paiement (DSP2)* ».

## Une montée en compétences progressive des nouveaux arrivants

Les acronymes de la finance, comme l'incontournable « DSP2 », sont bien sûr méconnus dans le monde de l'informatique. Pas de panique : il n'est pas nécessaire de connaître ces réglementations, ni même le monde de la finance, pour postuler chez Lemon Way. Et pour cause : « *Nous organisons une montée en compétences graduelle de nos collaborateurs. Les nouvelles recrues sont guidées pour découvrir le fonctionnement de nos modules en travaillant d'abord sur des tâches simples* » rassure Gilles Réant.

*« Leur rôle est ensuite d'évoluer au bout de quelques mois vers des tâches plus complexes et plus créatives lorsque les métiers sont progressivement assimilés ».*
L'équipe RH veille également au bien-être des nouveaux collaborateurs. Le terme de « collaborateur » est d'ailleurs banni : en interne, on parle de « Lemon Hero ». *« Nous venons de mettre en place un parcours d'intégration pour les nouveaux arrivants »* explique Carmen Ionita. *« Nous sommes là pour accompagner les Lemon Heroes à chaque étape de leur vie professionnelle. Nous avons ainsi entièrement repensé l'expérience des collaborateurs et des candidats ».*
Pas d'angoisse en vous rendant à votre entretien d'embauche : vous serez bien traité !

## Programmeurs C#, .Net, Vue.js : on vous déroule le tapis rouge !

La question est bien sûr de savoir quelles compétences sont recherchées. *« Nos développeurs back-end évoluent dans un environnement full Microsoft (C#, .Net) et ceux davantage axés front-end utilisent le framework Vue.js »* explique Mohamed Otmani. Une bonne connaissance de ces langages et des outils associés est donc requise pour intégrer Lemon Way. *« Nous sommes généralement à la recherche de personnes ayant déjà une première expérience de 2 ou 3 années dans le domaine de la programmation »* précise-t-il.
Naturellement, les rémunérations proposées aux collaborateurs sont compétitives sur le marché. *« Nous offrons à nos développeurs informatiques une rémunération qui s'ajuste à leur niveau d'expérience. Aujourd'hui, ils sont rémunérés uniquement sur une base fixe mensuelle, mais nous étudions la possibilité de mettre en place la rémunération variable pour certains postes qui pourraient correspondre à ce type de rétribution »* détaille Carmen Ionita.
Les profils internationaux sont accueillis à bras ouverts : *« Nos collaborateurs sont déjà issus de 15 nationalités différentes. Nous sommes attachés à notre ouverture culturelle et nous souhaitons la développer »* souligne Mohamed Otmani.

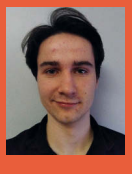
## Une ambiance startup toujours au rendez-vous !

On l'aura compris : tout en étant pionnière dans le paysage des Fintechs, Lemon Way n'a rien perdu de son identité de startup. *« Lemon Way garde au cœur de son ADN un esprit dynamique, entrepreneurial et convivial »* souligne Carmen Ionita. *« Ce ne sont pas les occasions qui manquent pour se retrouver et faire la fête, que cela soit dans un cadre sportif, culinaire ou lors d'afterworks ».*
L'esprit startup se retrouve également au quotidien dans les méthodes de travail : *« Il y a une vraie culture d'entraide et de solidarité chez Lemon Way »* souligne Gilles Réant. *« Par exemple, lorsqu'une équipe a un besoin technique spécifique correspondant aux compétences d'une autre personne, celle-ci peut être amenée à changer de groupe de travail pour partager ses connaissances là où elles sont les plus utiles. Nous mettons beaucoup l'accent sur la flexibilité et le partage des savoirs, ce qui passe aussi par des roulements d'équipes réguliers pour permettre à chacun de travailler sur des sujets variés et d'apprendre de nouvelles choses ».*
Cette flexibilité, qui se traduit également par la possibilité d'effectuer du télétravail, est appuyée par le département des ressources humaines. *« Les parcours professionnels en interne sont pensés pour ouvrir des perspectives d'évolution à tous. Pour nos développeurs informatiques, il est tout à fait possible d'effectuer des mobilités inter-équipes, de l'équipe Développement vers l'équipe Production, et vice versa »* explique Carmen Ionita. *« À l'avenir, nous souhaitons aussi faire évoluer nos développeurs vers des postes de management ou bien de Lead Developer et d'Architecte ».*


## Un avenir envisagé avec sérénité

Avec l'appui de nouveaux collaborateurs, Lemon Way envisage son avenir avec sérénité. *« Après deux belles levées de fonds réussies en 2018 et 2019, Lemon Way est une FinTech qui ne cesse de grandir »* sourit Carmen Ionita. L'avenir de Lemon Way s'annonce à la fois français, européen et international, avec déjà plusieurs implantation en Europe. Il ne reste désormais plus qu'à partager ces projets d'avenir avec de nouveaux Lemon Heroes !

Vous souhaitez postuler ? N'hésitez pas à envoyer votre CV à l'adresse **recrutit@lemonway.com** !



Maxime Ellerbach
*Lycéen*
Github : https://github.com/Maximellerbach

# Détection de visages avec le Maix Dock de SiPeed

*Le Maix Dock développé par Sipeed est une petite board destinée à faire de l'embarqué. Elle possède un processeur un peu particulier car il embarque un FPU (Floating point unit) et des accélérateurs de convolutions !*

## Un peu plus sur le hardware

Côté performance, on ne peut pas vraiment le comparer à un Arduino ou à un Raspberry Pi, mais plutôt à des cartes comme des OpenMV qui sont orientées computer vision intégrant directement une caméra sur le board.

Pour un prix moindre, le maix Dock embarque une caméra certes moins performante, mais possède une puissance de calcul plus grande et une mémoire plus conséquente lui permettant de charger des modèles de DeepLearning et de faire les faire marcher en temps réel.

Son KPU (Key Processing Unit) est intéressant, car SiPeed a choisi d'embarquer un FPU de seulement 8 bits permet de faire des calculs plus rapides, mais en contrepartie moins précis que des flottants de 16 ou 32 bits. De plus, les accélérateurs matériels de convolutions lui donnent un avantage certain pour le DeepLearning et notamment pour les CNN (Convolutional Neural Network) très utilisés pour de la reconnaissance d'image. On obtient donc une board à la fois puissante, mais aussi performante et qui ne rentre pas forcément dans une catégorie prédéfinie.

**Note**

*la webcam et l'antenne sont livrées avec la board*

## Installation et flash de l'OS

En suivant le guide d'installation de SiPeed: https://maixpy.sipeed.com/en/get_started/upgrade_firmware.html, tout s'est bien passé. À noté tout de même que toutes les cartes SD ne sont pas compatible avec la board, j'ai dû en tester plusieurs avant d'en trouver une compatible.

Pour mettre à jour ou installer le firmware, il vous faudra télécharger un outil pour flasher l'image sur la board. Cet outil en question est assez simple à utiliser et vous permettra par la suite de flasher des modèles dans la mémoire.

Pour le télécharger : https://github.com/sipeed/kflash_gui

Après l'avoir téléchargé, il vous faut choisir une image à flasher, elles sont disponibles ici : http://dl.sipeed.com/MAIX/MaixPy/release/master/. Pour ma part j'ai choisi une des dernières avec le support d'affichage graphique indiqué dans la version par « with_lvgl »

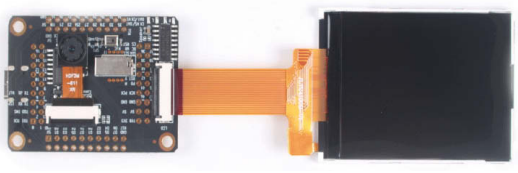
Pour flasher, il vous suffira de choisir l'image, le port COM où se trouve votre board, la vitesse d'écriture et l'endroit où écrire l'image dans la mémoire. Une fois paramétré, cliquez sur Download et hop ! 1

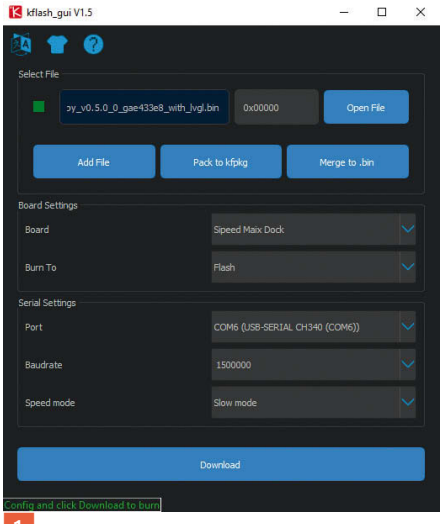
## Test, transferts de fichiers et exécution de programmes

Pour cette partie, deux options s'offrent à vous pour exécuter des programmes sur la carte. Vous pouvez utiliser le classique Putty en vous connectant au bon port COM, ou utiliser l'IDE de SiPeed spécialement conçue à cet effet (http://dl.sipeed.com/MAIX/ MaixPy/ide/).

La deuxième option s'avère être horriblement décevante, car pour transférer des fichiers il faut que la board soit connectée et liée à l'IDE cependant, à chaque transfert ou exécution de fichier, la board se voit déconnecter et je n'ai trouvé d'autre moyen que de le rebooter puis de le reconnecter à l'IDE. Quelque chose qui devrait en principe être réglé dans les prochaines versions de l'outil.

Pour ma part, je me suis vite redirigé vers Putty pour l'exécution des programmes et pour le transfert de fichier, j'ai simplement connecté la carte SD à mon PC, copié les fichiers dessus puis l'ait remise dans le board. Pour tester si tout fonctionne bien quoi de mieux que de faire un classique « helloworld » ?

| Board | OpenMV cam H7 | SiPeed Maix Dock |
|---|---|---|
| |  |  |
| Processeur | 1 cœur 32 bits RISC @216Mhz | 2 cœurs 64 bits RISC-V @400Mhz-600Mhz (overclockable) |
| Caméra | VGA (640x480) @60 FPS | QVGA (320x240) @ 30 FPS |
| RAM | 1Mo | 8Mo |
| KPU | FPU floats 16/32 bits | FPU floats 8bits + accélérateurs de convolutions @400Mhz-600Mhz (overclockable) |
| Stockage | 2Mo Flash + carte SD | 16 Mo Flash + carte SD |
| OS | MicroPython | MicroPython |
| IOT | Shield wifi à 30 $ USD | Antenne wifi |
| micro | Non | Oui |
| Prix | 65 $ USD | 25 $ USD |
| source | https://openmv.io/collections/products/products/openmv-cam-h7 | https://wiki.sipeed.com/ |




1

```
helloworld.py
1  print("Helloworld !")
```

Après avoir confectionné mon magnifique helloworld, je le transfert sur la SD et une fois branché au board, je me connecte avec Putty et l'exécute avec « import helloworld », cependant, je suis fainéant et j'aime que les choses s'exécutent au démarrage… Je crée un fichier « boot.py » dans lequel j'importe mon helloworld, même processus pour le transfert de fichier, je reboot et hop le helloworld s'affiche !

```
COM6 - PuTTY
[MAIXPY]Pll0:freq:806000000
[MAIXPY]Pll1:freq:598000000
[MAIXPY]Pll2:freq:45066666
[MAIXPY]cpu:freq:403000000
[MAIXPY]kpu:freq:598000000
[MAIXPY]Flash:0xc8:0x17
open second core...
mallocmallocmallocmalloogc heap=0x80306770-0x80386770(524288)
[MaixPy] init end

Official Site : https://www.sipeed.com
Wiki          : https://maixpy.sipeed.com

Helloworld !
MicroPython v0.5.0 on 2019-11-29; Sipeed_M1 with kendryte-k210
Type "help()" for more information.
>>>
```

Bon… Maintenant, passons au niveau supérieur !

## Détection de visages

Pour l'utiliser dans notre programme, nous devons installer un modèle de détection de visage, ça tombe bien ! SiPeed nous en met un à disposition : http://dl.sipeed.com/MAIX/MaixPy/model/. Même processus que pour l'image de l'OS, on le charge dans l'outil de flash, on le burn. Une fois flashé on peut passer au code !
Comme toujours, on importe les modules que l'on va utiliser :

```
import sensor
import lcd
import KPU as kpu
```

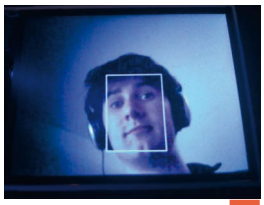
Ici, « sensor » permet d'utiliser la caméra, « lcd » d'afficher sur l'écran et « kpu » de charger des modèles et d'effectuer des prédictions avec.

```
task = kpu.load(0x300000)
anchor = (1.889, 2.5245, 2.9465, 3.94056, 3.99987, 5.3658,
    5.155437, 6.92275, 6.718375, 9.01025)
kpu.init_yolo2(task, 0.7, 0.7, 5, anchor)

lcd.init(freq=15000000)
lcd.direction(lcd.YX_RLDU)
sensor.reset()
sensor.set_pixformat(sensor.RGB565)
sensor.set_framesize(sensor.QVGA)
sensor.skip_frames(time = 1000)
```

On charge le modèle se trouvant à l'endroit 0x300000 dans la mémoire flash puis on initialise les différentes tailles de « bounding box » utilisé pour « yolov2 », ce modèle a été entraîné avec ces paramètres, si on les change, on se retrouvera avec des box qui sont mal équilibrés (détecte mieux de loin que de près ou autre). Enfin on initialise l'écran et la caméra.

```
p = "/sd/images/"
while(True):
  img = sensor.snapshot()

  try:
    pred = kpu.run_yolo2(task, img)
    if pred:
      rects = []
      for i in pred:
        rects.append(i.rect())
        it += 1

        tosave = img.copy(i.rect())
        tosave.save(p+str(it)+".JPEG")
        del tosave

      for r in rects:
        img.draw_rectangle(r)

  except OSError as e:
    img.draw_string(2,2, str(e), color=(0,128,0), scale=2)
    pass

  lcd.display(img)
  del img
```

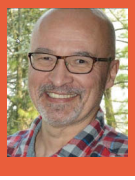
À chaque itération, on capture une image puis dans un « try » pour éviter que le programme crash en cas d'erreur on prédit à partir du modèle les différentes « boxes » puis on parcourt chaque box et on en sauvegarde la région de l'image concernée, on dessine un rectangle pour chaque box détecté sur l'image capturée, enfin, on affiche l'image ! Testons tout ça !
Après environ 10 secondes de chargement, le programme se lance et voici ce qui s'affiche à l'écran : 2
Le modèle est assez robuste et tourne convenablement à environ 10 itérations par seconde, le plus long lors de l'exécution est l'enregistrement de l'image, bien que petite, elle fait souffrir les deux cœurs du processeur, sans la sauvegarde de l'image, on atteint facilement 20 itérations par seconde



2

## Conclusion, avantages, défauts

Malgré sa petite taille et son prix relativement faible, cette board est impressionnante d'un point de vue optimisation avec un KPU puissant. Elle est capable de faire tourner des modèles sur un processeur qui reste fondamentalement peu puissant. En revanche, j'ai ressenti un réel manque de documentation en anglais (c'est un problème pour de nombreuses cartes, NDLR). Voulant trouver un moyen d'overclocker la board, je suis passé par la documentation chinoise qui elle, bien que difficile à comprendre était complète ! Si vous parlez chinois, je suis sûr qu'ils apprécieraient de l'aide ! Sinon, Bing translator est ton meilleur ami. :-)
Côté performance, j'aurais aimé quelque chose avec un processeur plus véloce tout de même, qui ressemblerait plus à un Raspberry Pi (ou un équivalent). Ça reste un bon produit très sympa pour bricoler de simples applications rapidement. Si vous aimez bidouiller avec des boards sans réel but à part celui de s'amuser : elle est géniale ! Si vous cherchez une carte capable de faire du Deep learning je vous recommanderai plutôt un Raspberry Pi (ou équivalent) ou un Jetson Nano selon vos besoins.



Francois Lionet
**Founder - AøZ Studio**
https://aoz.studio

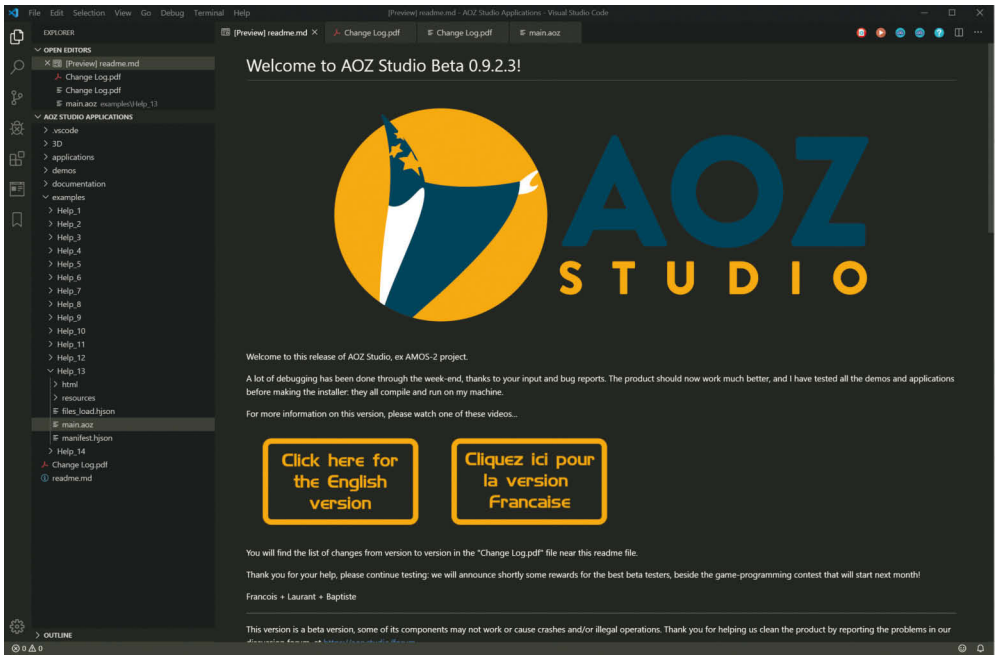
# La magie d'AOZ : le retour du Basic

*Je m'appelle Francois Lionet, et je vais vous présenter dans cet article ma dernière création, AOZ Studio, qui représente la synthèse de tout ce que j'ai appris en 39 ans de programmation, et surtout, la raison d'être de ce projet : apprendre à coder d'une manière simple.*

J'ai attrapé le virus de la programmation en 1981 et développé au long de ma carrière plusieurs moteurs de jeux sur différentes plateformes : STOS sur Atari ST, AMOS sur Commodore Amiga, puis Klik and Play sur PC Windows, suivi de Multimedia Fusion et enfin Clickteam Fusion avec Yves Lamoureux. Et aujourd'hui, j'ai créé la société AOZ Studio avec mes partenaires dont Laurant Weill, ex-fondateur de Loriciels.

## Fabriquer des jeux pour apprendre à coder

La caractéristique principale des produits créés au long de ma vie était de permettre aux gens ne connaissant pas l'informatique et la programmation de pouvoir réaliser, à l'aide d'un langage simple de syntaxe Basic (STOS et AMOS) ou graphique (Klik and Play) leur propre jeu. Avant d'être des outils d'éducation au codage, ces produits étaient conçus comme de véritables moteurs de jeux, écrits en langage machine pour une bonne rapidité d'exécution, et présenter d'une manière simple les notions complexes associées au codage de tout jeu.
J'ai connu les ordinateurs au temps où ils étaient simples. Quelques secondes après le boot, le curseur du Basic intégré apparaissait, et on pouvait se lancer. C'est cette simplicité qui m'a permis, n'ayant jamais touché un ordinateur de ma vie, de réaliser mon premier programme sans bug moins de 5 minutes après avoir mis en route mon Superboard II en 1981, sous la forme d'un simple Print "Hello world".
Tout de suite récompensé sans avoir fourni de gros effort, cela m'a immédiatement engagé à poursuivre l'exploration de ce qui allait devenir rapidement une passion... car finalement, "cela n'avait pas l'air si compliqué" de programmer.
Je ne suis pas le seul dans ce cas. Un grand nombre de professionnels des TI de mon âge ont commencé comme cela, et ont finalement tout appris tout seul dans leur chambre.
La simplicité des machines associée à celle du Basic permettait de découvrir la programmation à son rythme, tout en étant constamment récompensé par des résultats de plus en plus bluffant à l'écran et l'appréciation tintée d'admiration de son entourage.
Comme le dit le proverbe, "c'est le premier pas qui compte".




## L'enfer du débutant d'aujourd'hui

Apprendre à coder aujourd'hui pour un débutant absolu est "une autre paire de manches". Se pose d'abord le choix du langage. Lequel choisir ? Si, comme beaucoup d'enfants et d'adolescents le désirent, on veut faire un jeu, on se retrouve devant une pléthore de moteurs divers et variés, en programmation graphique ou pas, tous utilisant différent langages.
Programmation graphique et évènementielle, telle que l'offrait Klik and Play en 1993 ... Scratch? Game Maker? Construct? RPG Maker? Fusion? AppGamekit? Etc.
Coder ? Mais avec quel langage et quel moteur... Java ? JavaScript ? C# ? C++ ? Lua ? Python ? Unity ? Unreal ? Godot ? Three.js ? CryEngine ? etc.
Faire des recherches Google ne fait qu'accroître la confusion : chacun défend le moteur et le langage qu'il utilise comme étant le meilleur de tous et critique (souvent) les autres. Une fois le moteur et le langage choisi, il reste à installer les outils de développement, ce qui peut aller du plus simple au plus compliqué et prendre jusqu'à plusieurs heures. Et bien entendu, il faut choisir une IDE, pour enfin pouvoir commencer à coder.
À la complexité du choix de l'outil et du langage et de la phase d'initialisation de tout logiciel avant de pouvoir sortir quelque chose à l'écran, se rajoute la difficulté des langages eux-mêmes... Qu'est-ce qu'une accolade, à quoi ça sert ? Pourquoi un point et pas une flèche ? Que veut dire cette erreur ? Pourquoi ça ne compile pas ?

Ça veut dire quoi "this" ? Pourquoi ça plante ? C'est malheureusement un fait : un grand nombre d'aspirants programmeurs se découragent après plusieurs jours de bataille contre la machine, et retournent jouer à la console en se disant que décidément, ils ne seront jamais programmeurs. Abandonnant ainsi leur rêve de créer pour ne pas rester un simple utilisateur.

## La confusion également dans les écoles

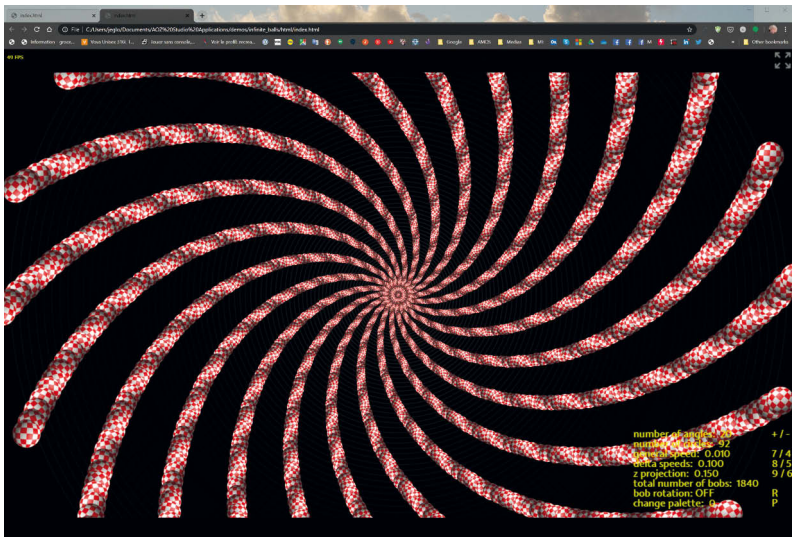
La France est célèbre pour ses grands "plans" d'enseignement de l'informatique, à chaque fois promus à grand renfort de communication politique. Les gens de mon âge se souviennent du fameux "Plan Informatique pour Tous" lancé en 1985 par Laurant Fabius et de son échec quelques années plus tard. Miné par un mauvais choix de matériel, un manque de formation des enseignants, et une totale incompréhension de la part des politiques de l'outil informatique et des désirs des enfants, ce plan de plusieurs milliards a été rapidement abandonné.

En Angleterre, à la même période, était diffusé sur la BBC "The Computer Programme". L'émission présentait les ordinateurs sous l'aspect du jeu et de l'interactivité, et utilisait une machine de grande qualité, le "BBC Micro" fabriqué par Acorn. En seulement un an de diffusion, cette émission exceptionnelle encore dans les mémoires est responsable de l'avance du pays dans le domaine du jeu entre 1980 et 2000.

Les choses se sont un peu éclaircies dans notre pays depuis ces premiers pas hasardeux et les mauvais choix initiaux. S'est développé au fil des années un véritable cursus d'enseignement de l'informatique. Mais, cet apprentissage, qui commence dès le cours élémentaire, présente en premier le côté rébarbatif et compliqué des ordinateurs à des enfants qui ne pensent qu'à jouer à la console. Pour preuve, voici l'extrait, tiré de Wikipedia du programme destiné aux enfants de 8/10 ans dans nos écoles aujourd'hui :

*En cycle 2, […] Ils se familiarisent avec le traitement de texte : ils apprennent à supprimer, déplacer et dupliquer des paragraphes et à utiliser un correcteur orthographique.*

Commencer à apprendre les ordinateurs aux enfants par le traitement de texte est

certainement la manière de les dégoûter de l'outil. Alors que je sais, par le retour que j'ai, qu'un enfant même de -10 ans est parfaitement capable de créer des jeux.

## Basic à la rescousse !

Présent sur 100% des machines dans les années 80, le Basic a progressivement disparu pour n'être plus aujourd'hui qu'anecdotique. Cette disparition est selon moi, due à plusieurs facteurs :

- il était associé à une certaine "lenteur" d'exécution, certes justifiée pour les premières machines dont les miennes, car il s'agissait d'interpréteurs, mais sans fondement plus tard avec les compilateurs.
- la fameuse instruction GOTO, symbole du Basic, si simple à appréhender pour le débutant est devenue le symbole du "mauvais programmeur" et du "code spaghetti". Elle est pourtant disponible dans des langages comme le C, et se révèle bien pratique dans certains cas… à condition de ne l'utiliser qu'avec parcimonie.

C'est avec Visual Basic, que Microsoft a sans le vouloir, achevé le langage. Visual Basic était un Basic très évolué et événementiel, qui permettait de réaliser des applications professionnelles. Or les milieux pros étant ce qu'ils sont, clamer que programmer en Basic ne faisait pas bon effet sur un CV. Et juste le nom, "Basic", suffit à dénigrer le langage. Un langage professionnel aurait plus de chance de succès s'il s'appelait "Compliqué"…

Quel dommage ! Car il représente le langage idéal pour débuter, et ce, quel que soit l'âge.

Même s'il ne comprend pas un mot d'anglais, un enfant de 8 ans comprendra la ligne suivante et ce qu'est un programme après avoir tapée et la voir fonctionner :

```
Print "Bonjour!"
<RUN>
Bonjour!
```

Puis cinq minutes plus tard après quelques explications de la part du professeur :

```
Input "Quel est ton nom ?"; NOM$
Print "Bonjour "; NOM$
<RUN>
Quel est ton nom? Francois
Bonjour Francois
```

En dix minutes, montre en main, le Basic vous permet de comprendre la notion de programme et de variable, sans erreur et sans recherche, avec une syntaxe simple proche du langage parlé.

## Programmer comme on parle

C'est la caractéristique principale du langage Basic : on programme comme on parle. Alors que les langages actuels essaient de réduire au maximum le nombre d'instructions et de définitions de structure, passant toute la complexité sur les librairies, le langage Basic comprend un grand nombre d'instructions, chacune portant un nom clair et proche de la langue parlée.

Pour un enfant, apprendre à programmer en Basic revient à apprendre le nom des instructions, tout en les associant à une action, comme on apprend une récitation. Et comme pour les phrases, c'est l'association des mots dans le bon ordre, et simplement cela, qui forme la phrase, la routine, l'application.

On raconte des histoires aux enfants. Avec le Basic, ils se racontent eux-mêmes des histoires, avec un vocabulaire magique dans lequel un seul mot fait apparaître une image sur l'écran, et un autre la fait bouger ou exploser.

## Le retour de la simplicité

AOZ est la re-écriture avec les technologies modernes JavaScript et HTML5 de STOS et AMOS, qui furent de gros succès en Angleterre et dans le reste de l'Europe -sauf peut-être en France- entre 1987 et 1993.

Il s'agit d'un transpilateur et non pas un compilateur, qui prend en entrée du code de syntaxe Basic et le converti en JavaScript, produisant des applications prêtes à fonctionner dans le navigateur sans serveur, ou à être wrappée par des systèmes tels que Cordova en exécutables pour machines mobiles ou PC. Il est donc possible avec le même code source d'exporter sur toutes les machines existantes.

Le moteur utilise Three.js pour le rendu, et de nouvelles instructions pour fabriquer des jeux en 3D ont été ajoutées. Il supporte aussi les shaders, les Google Fonts, les résolutions modernes et peut s'interfacer avec toutes les technologies du monde JavaScript et/ou node.js, tant back-end que front-end, APIs Rest, REACT, mySQL, etc. Et fonctionne évidemment à la vitesse de JavaScript, c'est-à-dire proche de celle du C.

Le transpilateur produit un code clair et lisible qui met directement en relation le code Basic avec le code JavaScript, et offre une option d'obfuscation. L'utilisateur peut, en ouvrant une simple accolade, intégrer du JavaScript directement dans le code Basic :

```
A = 0
{
    // Nous sommes en JavaScript
    this.vars.A = 1;
}
Print A
1
```

Cette fonctionnalité permet aux étudiants de faire la transition progressive de Basic vers JavaScript et découvrir ce langage, porte d'entrée du monde pro.

Le produit comprend directement les applications sauvegardées sur les machines de l'époque, et les convertis en une structure moderne avec les éléments constituants séparés, le code source en ASCII, les images en PNG, les sons en WAV, etc.

La première version, disponible en mars prochain, utilise Visual Studio Code comme IDE. La version suivante, prévue pour juin, comprendra notre propre IDE programmé en AOZ lui-même, et sera disponible en version locale et en ligne, ainsi que de nombreux accessoires pour éditer et animer des sprites, des sons, des maps, etc. formant ainsi une véritable game-engine avec tous les outils nécessaires pour fabriquer des jeux.

Sur notre roadmap, se trouve l'évolution du transpilateur, avec l'ajout d'export vers d'autres langages tels le C# et l'interfaçage avec Unity pour accéder au marché des consoles, C/C++ pour utiliser Unreal, mais également pour la programmation sur ESP32 et Arduino.

Je suis persuadé que puissant ne doit pas forcement dire compliqué. Puissant doit plutôt signifier profond, et c'est l'utilisateur lui-même qui doit décider de creuser ou pas pour utiliser cette puissance. Et ne pas devoir comprendre des montagnes de notions pour n'afficher qu'un simple Pac-Man. C'est notre but avec AOZ et AOZ Studio : que tout le monde puisse devenir des magiciens du code. Que tout le monde puisse découvrir la programmation par le jeu et la pratique, et puisse ensuite passer à d'autres langages pour entrer dans le monde pro par la grande porte.

Site officiel : https://aoz.studio


## CONCOURS SPÉCIAL
## LANCEMENT D'AOZ STUDIO

*Vous savez développer, vous désirez apprendre, vous voulez vite réaliser le jeu ou l'app de vos rêves ? AOZ STUDIO lance l'outil dont vous rêviez…*
*Vous pouvez faire partie des magiciens d'AOZ et pour découvrir participer au concours.*

**Objet du concours : réaliser le jeu ou le programme de votre choix (attention à être bien propriétaire des droits)**

**Fin du concours : 30 Mars 2020**
**Remise des prix : 20 avril 2020**

**À gagner :**

**1er prix :** 300€ + licence à vie d'AOZ Studio avec 2 ans de support + Abonnement 1 an à Programmez!
**Du 2e au 4e prix :** Licence à vie d'AOZ Studio avec 2 ans de support + Abonnement 1 an à Programmez!
**5e au 6e :** Licence d'1 an d'AOZ Studio + licence d'1 an à David Gamecodeur
**7e au 9e :** Licence de 6 mois d'AOZ Studio + licence de 6 mois à David Gamecodeur.

Pour participer télécharger (gratuitement) la version bêta sur www.aoz.studio

Pas d'inscription, mais faîtes nous savoir que vous démarrez par un simple mail avec votre projet résumé sur support@aoz.studio cela nous fera plaisir de vous suivre.

En participant à ce concours vous comprenez que chaque participant ne peut gagner qu'une fois, que le ou les auteurs du programme en reste propriétaire, décide(nt) seul(s) si il y a une rémunération associée et autorise AOZ Studio à publier et communiquer sur ce programme.

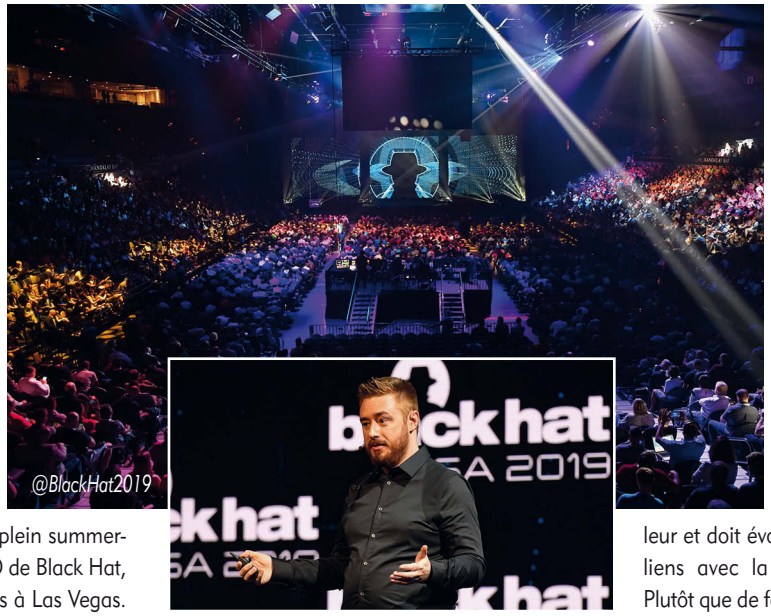
**C'est le moment de devenir magicien…**
La team AOZ Studio.

**En partenariat avec Programmez! Et David Gamecodeur**

Véronique Loquet
Twitter @vloquet

# Black Hat : Dino Dai Zovi, hacker émérite new-yorkais, milite pour la coopération et le DevSecOps

Sur la scène internationale de l'infosec, Dino est une star… ainsi va l'infosec circus. À la tête de la sécurité chez Square, le quotidien de ce chercheur est fait de reverse engineering, de code, d'exploits, d'audits et de recherche disruptive. Il a ouvert l'édition 2019 de la Black Hat dans une arène bondée de plus de 20 000 professionnels de la cybersécurité. Le show démarre, son rock et spotlights saturés, Jeff Moss le boss de Black Hat s'impose dans le décor, il se sent « comme à la maison », nous aussi. Nous sommes le 5 août, en plein summer-parties, le créateur de Def Con, CEO de Black Hat, réussi l'exploit d'un 22e rendez-vous à Las Vegas. Des années de rassemblement qui ont renforcé la communauté internationale et une conférence démente qui conserve son inspiration et une capacité à fédérer hors norme.




Depuis au moins 20 ans qu'on se dit que les développeurs sont sensés intégrer la culture sécurité, on a l'impression que le sujet est éculé. Alors que s'est-il passé pour que cette keynote d'ouverture à l'allure vintage donne l'impression d'innover ?

En projetant des images rétro de sa vie de hacker des années 2000 (sic), Dino Dai Zovi parle de bouleverser la culture sécurité pour encourager l'engagement de chacun. À l'université il passe tout son temps libre à trouver des failles de sécurité et à s'entraîner à les exploiter, plus tard il concourt au Pwn2Own lors de la conférence CanSecWest, une époque où sa posture dans la sécurité est essentiellement offensive.

Or il comprend que l'intégration des besoins logiciels, et par essence ceux du développeur, sera plus bénéfique à l'impact de son propre job pour faire face aux défis de la sécurité. Cependant les années passent et la culture cybersécurité n'est toujours pas suffisamment ancrée dans les organisations. Pour Dino la révélation arrive en 2014, lorsqu'il constate que les ingénieurs sécurité chez Square codent comme tout le monde. Dès lors sa perception de l'influence de la sécurité dans l'entreprise prend une nouvelle dimension, l'empathie favorise la collaboration.

Il évoque trois grandes étapes pour se prémunir efficacement et sécuriser au maximum un produit. La première doit intervenir en amont de la conception logicielle, la seconde doit permettre d'automatiser l'audit de code tout en poursuivant le dialogue interne, enfin, il s'agit de diffuser et d'adopter la culture sécurité dans un cadre collaboratif et ouvert. Cette approche qui semble si évidente doit permettre à tous de travailler ensemble, en co-responsabilité et tout au long du cycle de développement. Pour Dino, il ne doit pas y avoir de frontière entre les développeurs et les ingénieurs en sécurité. Puisque la sécurité est trop souvent vue comme un frein à l'innovation, il propose aux équipes de sécurité de commencer par dire « Oui, comment puis-je vous aider ? », et de s'attacher à soutenir les besoins exprimés. Selon lui l'attitude « Mister No » est contre-productive, elle traduit la peur et celle-ci n'est pas rationnelle. Cette peur est aussi un argument fallacieux qui incite à ne rien faire. Se concentrer sur les attaques 0day et manquer le reste n'est pas constructif. Moult pratiques issues du développement agile et du DevOps peuvent s'appliquer à celles de l'ingénierie en sécurité pour un maximum d'impact.

## « Tout le monde a un rôle à jouer dans la cybersécurité »

Selon Dino Dai Zovi, nous avons atteint le point de transition en matière de sécurité. La logique du pré carré entretenue par la communauté des chercheurs en sécurité a prouvé sa limite de valeur et doit évoluer vers l'opportunité d'élargir les liens avec la communauté des développeurs. Plutôt que de faire culpabiliser ces derniers quoi de plus bénéfique quand on peut directement contribuer à leurs côtés ?

Dans une ère nouvelle où la recherche de failles se monnaie, l'investissement dans une sécurité renforcée et le rapprochement des équipes opérationnelles apportent une valeur ajoutée significative à l'entreprise. Le management a parfois encouragé une approche en silo et donné la priorité aux délais de production. Si le DevOps a révolutionné la manière de répondre aux besoins changeants des clients sans sacrifier la productivité, la sécurité en continue n'a pas été intrinsèque au mouvement, ni le DevSecOps une priorité. La sécurité est rarement au cœur des préoccupations des équipes opérationnelles qui se passent parfois d'observer toutes les mesures de sécurité pour atteindre les délais imposés.

Une réflexion profonde en faveur du changement doit s'opérer, car c'est en permettant une large diffusion de la culture sécurité dans les entreprises que chacun sera conscient de sa responsabilité et que la transformation sera réelle. Être agile, collaborer, communiquer sont essentiels pour faire face aux menaces actuelles. Les équipes de sécurité qui adoptent cette approche réduisent les risques tout en minimisant l'impact sur la productivité globale.

# YesWeHack lance une plateforme de formation autour du Bug Bounty

***Le savoir c'est le pouvoir ! Et YesWeHack l'a bien compris. La start-up française, leader du Bug Bounty en Europe, a décidé de transmettre tous les enseignements qu'elle a tiré de son expérience en matière de détection et de divulgation de failles de sécurité à travers le lancement de sa plateforme éducative, YesWeHack EDU.***

YesWeHack EDU s'adresse ainsi aux promotions cybersécurité des écoles d'ingénieurs et plus largement à l'ensemble des promotions spécialisées IT (Big Ddata, développement, etc.) qui veulent accélérer le partage de datasets (jeu de données) de qualité. L'approche pédagogique de YesWeHack EDU vise d'abord à encourager l'émulation via la gamification et l'implication de chaque élève dans la sécurisation de son institution. Elle ouvre aussi et surtout des perspectives aux futurs développeurs pour aller vers des spécialisations prometteuses : DevSecOps, security-by-design, etc. Afin de démontrer le fonctionnement de sa solution, YesWeHack nous avait d'ailleurs conviés à une mise en situation d'une session de formation et d'apprentissage en la présence d'élèves de l'École Supérieure de Génie Informatique.

« Avec le nombre croissant de menaces et d'attaques auxquelles on se trouve aujourd'hui confrontés, la cybersécurité est devenue un enjeu à la fois économique et sociétal. Malgré l'importance que prend la cybersécurité, il est toutefois dommageable de voir que le secteur souffre d'un déséquilibre entre l'état de la menace et les capacités de défense du marché, » déclare Guillaume Vassault Houliére, CEO et Cofondateur de YesWeHack. Il rappelle qu'aujourd'hui près de 3 millions de postes sont à pourvoir dans le monde en matière de cybersécurité, dont plusieurs milliers en France.

C'est pour remédier à cette situation que YesWeHack a lancé sa plateforme. L'idée derrière YesWeHack EDU est de permettre aux futurs acteurs du Bug Bounty de se former sur des cas et des environnements qui représentent la réalité et surtout d'adopter plus facilement les bonnes pratiques du Bug Bounty. « Aujourd'hui, ce qui fait la valeur du Bug Bounty, ce sont les rapports et les données que nous générons. Nous devons impérativement apprendre aux nouveaux talents à créer des informations de qualité pour avoir des sources d'informations les plus riches possible », déclare Damien Lecoeuvre, CTO de YesWeHack. Et les étudiants ayant commencé à travailler sur la plateforme abondent dans ce sens. « C'est important pour nous de pouvoir nous confronter facilement à des solutions qui sont semblables à ce que nous retrouverons dans des environnements professionnels. En outre, YesWeHack EDU lève de nombreux freins que nous avions précédemment quand nous voulions accéder à des plateformes de Bug Bounty », explique Aurélien Delagarde, étudiant de l'ESGI en quatrième année.

Aussi bien les étudiants que les enseignants présents à la démonstration saluaient l'apport pédagogique de YesWeHack, mettant en avant les aspects collaboratifs de la plateforme. « Quand on veut se lancer dans le Bug Bounty, on est souvent tout seul sur les plateformes au début. Avec YesWeHack EDU, on peut échanger et comparer nos rapports qui sont par ailleurs notés. Ceci nous permet de nous améliorer et d'assimiler plus rapidement les bonnes pratiques », explique Hadrien Bouffier, étudiant de l'ESGI. Il ajoute qu'auparavant, quand ils utilisaient des outils classiques de Bug Bounty, ils n'avaient pas de retour sur leurs rapports et n'avait donc aucune idée de si leur travail était bon et de comment s'améliorer.

### Aurélien Delagarde,

**Étudiant de l'ESGI en quatrième année.**

*« Aujourd'hui, on voit que le Bug Bounty est très important pour la détection de failles de sécurité sur les applications et les sites web. C'est pourquoi il est d'autant plus important que nous puissions nous former sur des outils appropriés. Sur YesWeHack EDU nous avons un vrai suivi de notre travail ce qui est bien plus pédagogique, d'autant que la plateforme nous permet d'uniformiser les données et les rapports que nous produisons. Nous pouvons facilement revenir sur ces derniers pour les comparer et s'améliorer. En outre, nous pouvons travailler sur des situations qui sont très semblables à ce que nous rencontrons dans la réalité sans avoir à passer par les plateformes officielles qui restent plus difficiles d'accès que YesWeHack EDU ».*

### Hadrien Bouffier,

**Étudiant de l'ESGI en troisième année**

*« Pour moi qui a un background de Blue team, le Bug Bounty est très complémentaire au pain test et au travail des Red team. L'avantage, c'est que c'est ouvert au monde entier et que dans le hacking il faut aujourd'hui pouvoir confronter différentes approches et différentes cultures. L'avantage de YesWeHack EDU c'est que ça nous apporte une plateforme clé en main. En plus de la plateforme, ce qui est important, c'est les programmes que peuvent créer les enseignants. Nous pouvons faire des instances de laboratoire de sécurité, créer des rapports et travailler dessus pour nous améliorer. Le rapport c'est d'ailleurs la grande valeur ajoutée du Bug Bounty. Sur YesWeHack EDU nous pouvons afficher tous les rapports et les commenter. Ça pousse à l'émulation entre nous et à nous améliorer. »*

En prenant les devants en matière de formation, les entreprises, les institutions et les acteurs de la formation peuvent rattraper le retard en matière de détection de faille et rentrer dans un cercle vertueux de partage de la connaissance. En mettant à disposition des étudiants des situations réelles sur lesquelles s'entraîner, les universités et les écoles amélioreront leur apprentissage. Les jeux de données générés par ces étudiants pourront ensuite être partagés et utilisés pour de nouveaux usages à destination des nouveaux métiers de l'IT, par exemple les data scientistes ou les DevSecOps. Et ainsi de suite.

**Denis Duplan**
*sociologue et développeur à ses heures.*
**Blog :** http://www.stashofcode.fr

# Root Me : racine-moi si tu l'oses !

*Lorsque je me suis intéressé à la sécurité informatique, je n'ai pas bien su par où attaquer la chose. Finalement, je me suis décidé à l'attaquer à sa racine, Root Me étant là pour ça.*

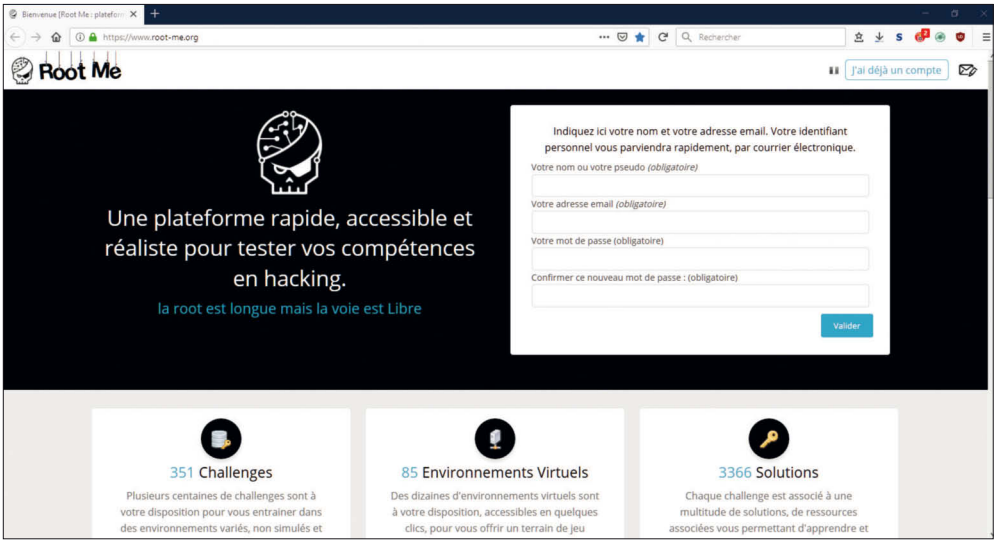
*La sécurité informatique est un (très) vaste domaine, c'est le moins que l'on puisse dire, d'autant plus que certains s'emploient à l'étendre joyeusement. Sans doute, dans la vie professionnelle, il faut savoir ravaler son domaine d'expertise pour l'accorder aux couleurs du temps, quitte à forcer un peu sur la peinture. Reste que cela ne va pas sans générer une certaine confusion. Pourquoi pas un « Grenelle du cul », avait proposé Roselyne Bachelot, lassée que l'on promette un Grenelle sur tout. On voit l'idée.*

Pour y mettre un pied — dans la sécurité informatique, pas au cul —, j'ai déterminé que je partirai de la base, c'est-à-dire de la technique, et qu'après avoir acquis un vernis, je ferai des choix. Toutefois, passé un certain temps à lire quelques documents et regardé quelques vidéos portant essentiellement sur les techniques d'intrusion analysées par des pentesteurs, il m'est apparu impossible de vraiment assimiler des connaissances sans m'adonner à une pratique assez intense, les sujets, même à ce niveau et dans cette spécialité, étant trop divers.

Pirater le réseau WiFi du voisin étant exclu, les MOOC m'ont semblé tout indiqués. Reste qu'après en avoir tâté un peu, je n'ai pas trouvé la motivation pour m'y investir plus que quelques heures. Le problème, c'est que ce n'était pas le challenge. On était loin de l'ambiance *demoparty*, « *elite rulez* », ou « *l337 rUl32* » comme on dit maintenant, et tout ce qui tire vers le haut parce que c'est compliqué, et parce que c'est reconnu.

Le salut devait venir d'ailleurs. Au hasard d'une rencontre — enfin, c'était une réunion traitant de cyberdéfense, donc le caractère fortuit est somme toute très relatif —, j'ai eu l'occasion de rencontrer quelqu'un pratiquant quelque chose de tout à fait passionnant, le forensics. Le gaillard, à qui j'ai exposé mon problème, m'a alors renvoyé sur Root Me.

Donc nous y voilà. Root Me, kézako ? C'est exactement ce qu'il vous faut si, jeune ou vieux soldat, il vous faut un peu sentir l'odeur de la poudre sur le champ de bataille pour vous mettre en marche. Root Me est un site web (https://www.root-me.org) (Figure 1) essentiellement gratuit, où vous




**1** *Le site de Root Me, avec son trop cool de logo.*

êtes invité à vous confronter à des challenges, gagner des points si vous les surmontez, et de là pouvoir plastronner un peu. Car pour ce que j'ai pu en juger, si vous jouez franc jeu, soyez certain que vous ne les aurez pas volés, vos points. **1**

Pour ma part, j'ai donc décidé de me concentrer sur les challenges de forensics. Serait-il possible de les faire tous ? Cela m'a semblé être une belle occasion d'en apprendre plus sur le domaine. Après avoir surmonté 23 de 25 challenges au fil des dernières semaines — il faut bien travailler et vivre par ailleurs, même si Root Me devient vite envahissant… – et engrangé 870 points — ce qui ne fait de moi qu'un *newbie* dans la hiérarchie indigène… –, l'heure m'a semblé favorable pour dresser un premier bilan.

## Ce que j'ai appris de Root Me

**Un premier intérêt** est d'apprendre à renouer avec l'ambiance devoir sur table. En effet, la première chose à faire, et à ne pas hésiter à refaire en cours de route pour ne pas s'égarer, consiste à bien lire l'énoncé et étudier les moyens mis à disposition.

En matière de challenges de forensics, l'énoncé est toujours d'un laconisme de prime abord déconcertant. Par exemple, le challenge *Vilain petit canard* :

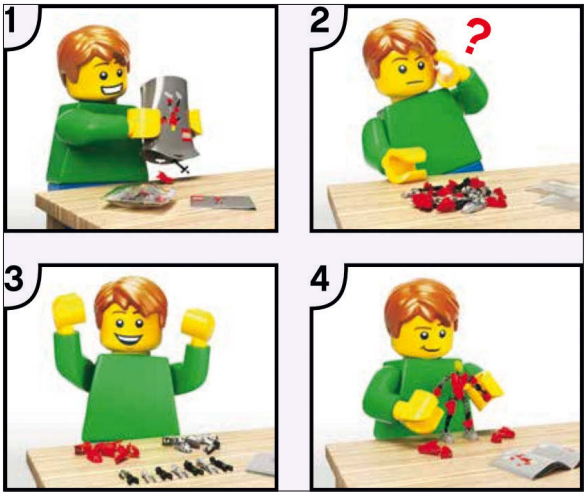
*Vilain petit canard*
*Comme un quack*
*L'ordinateur du DSI semble avoir été compromis en interne. Les soupçons se portent sur un jeune stagiaire mécontent de ne pas avoir été payé durant son stage. Une étrange clé USB contenant un fichier binaire a été retrouvée sur le bureau du stagiaire. Le DSI compte sur vous pour analyser ce fichier.*

Cet énoncé s'accompagne invariablement d'un ou de plusieurs fichiers à télécharger, qui constituent le matériau sur lequel il va falloir travailler, et éventuellement de références de documents renseignant d'assez loin sur des aspects techniques du challenge – par exemple, un simple manuel de l'outil votatility.

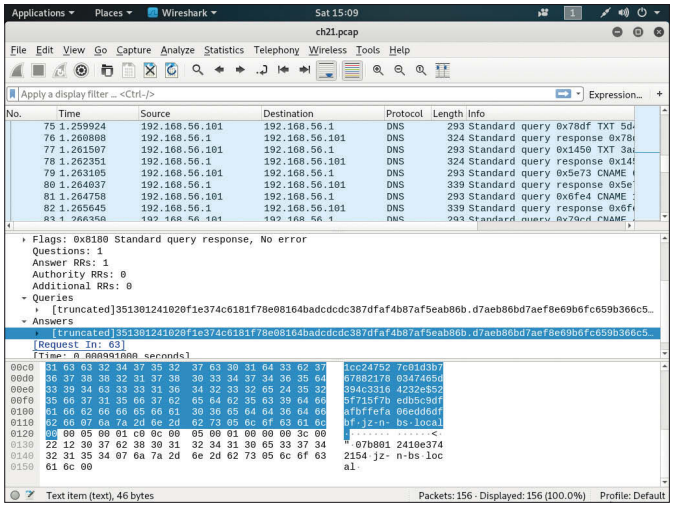
Il ne faut pas se laisser décourager pas le caractère aride du challenge, qui peut donner l'impression d'être débarqué sur une île

déserte, comme Link dans *Breath of the Wild*. En fait, la première chose à se rappeler, c'est qu'il faut lire l'énoncé. Certes, je viens de le dire, mais je ne suis pas certain d'avoir été bien lu. Quand je dis lire l'énoncé, c'est lire l'énoncé, en notant qu'il comporte trois parties : le titre, le sous-titre et l'exposé. C'est aussi lire les documents qui sont éventuellement référencés. En gros, c'est assez, comme se lancer dans un LEGO : il faut trier les pièces et lire la notice pour éviter de s'engouffrer sur une voie de garage (Figure 2).

Dans ses mémoires qu'il vient de publier, Edward Snowden souligne amèrement comment il a pu se prêter à une entreprise dont il a pu ultérieurement constater qu'elle était radicalement contradictoire avec ses convictions, tout simplement parce que focalisé sur son sujet, il avait perdu de vue *the big picture*.




2 *Tu aimes les LEGO ? Ben le forensics sur Root Me, c'est pareil.*




3 *Et moi, je dis que ces noms de domaine sont bizarres…*

De fait, prendre du recul n'est pas une pratique des plus répandues en informatique, ne serait-ce que parce que les projets sur lesquels l'on est mobilisé peuvent être très importants, que l'on ne peut pas être partout ni tout comprendre, car chacun a ses spécialités comme ses appétences. Si bien que du tout, l'on ne voit jamais qu'une partie – qui peut être plus que la somme, d'ailleurs. Aussi, sans que cela doive entraîner une crise de conscience comme chez Ed, mais simplement permettre de constater que l'on peut faire mieux en ayant conscience de ce que l'on fait, pratiquer ce type de challenge est utile.

**Un deuxième intérêt** : ils permettent d'en apprendre beaucoup sur des technologies, des techniques et des outils, pour autant que l'on sache saisir l'occasion.

Par exemple, je n'avais jusque-là guère trouvé la motivation pour installer une VM sous Hyper-V et y faire tourner Kali Linux, et au passage faire quelques lectures sur le fonctionnement d'un hyperviseur, jusqu'à un chapitre d'un manuel d'Intel pour bien comprendre comment cela se déroule au niveau du CPU.

J'aurais pu ne pas m'embarrasser et suivre un didacticiel sans me poser de questions, mais l'occasion était trop belle d'acquérir des connaissances sur les bases de la virtualisation, qui pourront très certainement me resservir en d'autres circonstances. Il en a été de même au fil des challenges pour toute sorte de sujets, du chiffrement à l'écriture d'un batch. Sans jamais aller jusqu'au fond des sujets, car ce serait impossible et ce n'est pas l'objet, chercher tout de même d'en faire un peu plus que simplement les effleurer.

À côté de ces explorations parfois annexes, il a bien fallu acquérir une certaine expérience sur des technologies, techniques et outils visiblement incontournables en forensics. Je pense à volatility, les outils de la suite TSK (The Sleuth Toolkit), ceux de la suite Sysinternals, WireShark et sa version en ligne de commandes tshark, et quelques autres (Figure 3).

Ces outils, j'ai dû non seulement apprendre à m'en servir, mais aussi apprendre comment il était possible de s'en servir de manière opportune et relativement efficace — la technique —, et parfois même comment parfois aller jusqu'à les adapter en fonction de mes besoins – la technologie. Par exemple, volatility étant écrit en Python, j'ai dû, car j'ai pu, rentrer dans le code de la commande vaddump pour la modifier afin qu'elle se ne crache que les traces des heaps d'un processus.

J'ai aussi dû parfois créer mes propres outils. Sans doute, ce n'était jamais que de petits outils puisqu'il ne s'agissait jamais que d'une fonction pour déchiffrer un bloc de données, mais cela ne m'en a pas moins donné une nouvelle occasion de me remettre à Python après quelques mois passés sur du JavaScript et de l'assembleur, donc de réapprendre des choses que j'avais oubliées, et d'en apprendre de nouvelles.

Pour en rester aux enseignements strictement techniques, il est clair que j'ai dû aussi rentrer dans des sujets qui étaient spécifiques aux challenges. Par exemple, quelle est la structure d'une app sous Android ? Autrement dit, que faut-il s'attendre à trouver dans un APK, et par quel bout faut-il s'en saisir pour comprendre ce que fait l'app ? Ou alors, comment est-il possible de mettre en place un keylogger sous Linux, et décrypter ensuite les structures des événements qu'il a permis d'enregistrer dans un fichier ? Ou encore, bien évidemment, que peut-on espérer trouver dans un dump de la mémoire d'une machine sous Windows, qui permette de comprendre ce qui était en train de se jouer à cet instant-là ?

On le voit, s'attaquer à un challenge de forensics, c'est aussi l'opportunité d'être bien contraint d'apprendre quelque chose sur des sujets informatiques auxquels l'on ne s'est jamais frotté, parce le dump que l'on vous balance est d'une version d'OS X sur Mac et que vous n'utilisez pas de Mac, ou parce que le programme que l'on vous jette est en Java et que vous ne codez pas avec, voire que dans un cas ou l'autre, voire les deux, au fond, vous détestez cela.

**Le troisième intérêt** : c'est… le forensics, ou du moins une certaine idée de ce qu'est le forensics, tel qu'on peut l'appréhender dans ces challenges. J'y reviendrai plus loin, en traitant de que je n'ai pas appris.

Ce qui ressort de ce forensics-là, c'est avant tout qu'il est passionnant. On progresse avec anxiété dans la jungle des données, surtout si l'on se met un peu la pression pour surmonter le challenge au plus vite, mais point trop pour ne pas rater les occasions d'apprendre que j'ai évoquées.

Après avoir effectué quelques repérages que l'on apprend à systématiser au fil des expériences, l'on trouve les premiers indices. Tiens ! l'arborescence des processus retournée par la commande pstree de volatility permet de constater que ce processus a donné naissance à un autre, alors que le lien entre les deux n'apparaît pas des plus évidents ? Ou alors, la dissection d'un PCAP par WireShark permet de constater que les réponses à des requêtes appelant des réponses lisibles se présentent sous une forme des plus ésotériques ? Hmm… il faut décidément aller voir cela de plus près.

Le caractère ludique des challenges est fortement accentué par l'esprit de compétition. Je l'ai évoqué rapidement au début de cet article, et il est temps d'y revenir, car c'est un aspect d'importance cardinale de Root Me : qui surmonte un challenge gagne des points.

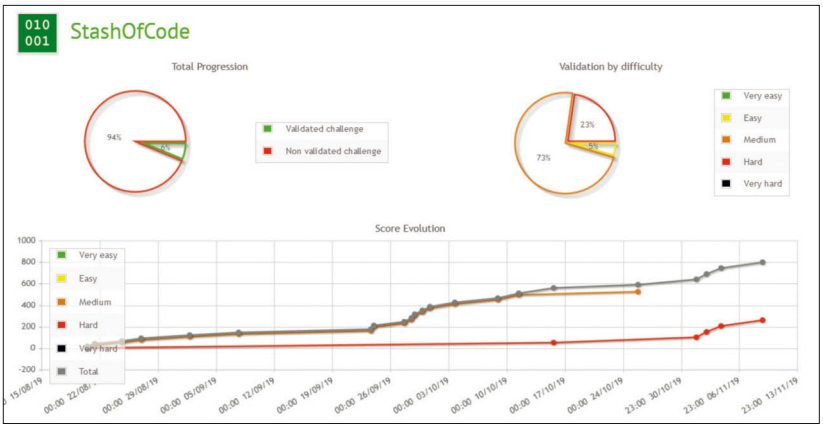
## La quête de points

Plus le challenge est difficile, plus il en rapporte, c'est évident. Pour ce qui concerne les challenges de forensics, le nombre de points à glaner va de 5 pour le premier, à 70 pour dernier. Si j'ai bien compté, il y a pour l'heure 25 challenges, ce qui laisse largement le temps de s'arracher les cheveux, la difficulté m'étant apparue réellement croissante.

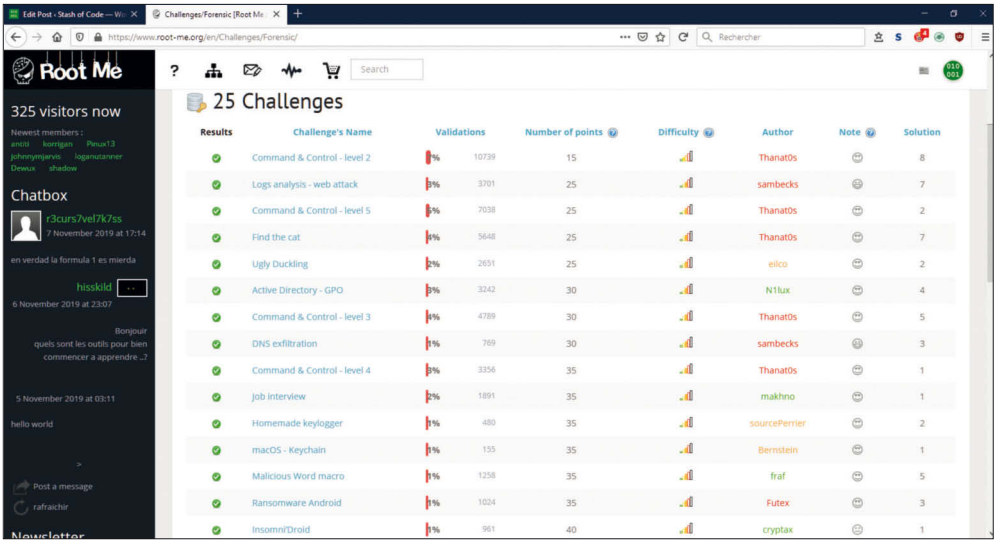
Au-delà des points, dans le profil de son compte, le rootmiste peut consulter des statistiques qui lui permettent de se faire une idée toujours réconfortante du chemin qu'il a pu parcourir s'il se souvient que quelques semaines plus tôt, il n'y entendait rien au forensics, à la cryptographie, ou que sais-je encore (Figure 4). Il peut aussi consulter la liste réjouissante des challenges où les vilains points rouges disparaissent, remplacés par de gentils points verts (Figure 5). Ainsi, en pratiquant les challenges, l'on s'entraîne naturellement à en relever, ce qui peut donner envie de participer un jour au CTF, l'acronyme de *Capture The Flag*, qui désigne une épreuve lors de laquelle il faut parvenir à trouver l'information enfouie dans les données avant d'autres, en temps réel.

Ce qui ressort aussi du forensics selon les challenges, c'est qu'il faut faire preuve d'une certaine méthodologie pour en tirer tout le bénéfice, et avant tout prendre des notes.

Prendre des notes est indispensable, car les investigations pouvant durer et les pistes sur




4 *La root est droite, mais la pente est forte, aurait dit « Raffarien ».*




5 *Même si pas écolo, on en vient à apprécier la verdure*

lesquelles s'engager se multiplier, l'on est bien content de pouvoir se rappeler où l'on en était quand il faut revenir en arrière ou simplement s'y remettre après une coupure. Personnellement, je trouve un intérêt à noter ce que j'ai fait et qui a bien marché, mais aussi ce que j'ai fait et qui n'a pas marché, histoire de ne pas refaire deux fois la même erreur. Quand un challenge me donne du fil à retordre, il me semble intéressant de prendre le temps de rédiger un petit résumé de mes notes, concentré sur ce qui a bien marché, histoire de prendre du recul.

Aussi, les notes permettent de disposer du matériau pour rédiger une solution et la proposer. Car les solutions sont accessibles sur Root Me, et il est toujours possible d'en proposer, pourvu qu'elle ne soit pas redondante avec une solution déjà publiée.

Bien évidemment, je ne me suis jamais risqué à cliquer sur le bouton permettant de les consulter avant d'avoir surmonté le challenge : en cas de blocage, il est recommandé de demander de l'aide sur les forums, ou simplement de consulter les réponses apportées à ceux qui en ont demandé.

Une fois un challenge terminé avec succès, je recommande de lire la ou les solutions proposées. En effet, si l'on veut bien se rappeler qu'il y a toujours plus d'un moyen de résoudre un problème, et que l'on n'a peut-être pas trouvé le plus futé, lire les solutions donne l'occasion de découvrir d'autres solutions, idées, techniques.

## Ce que je n'ai pas appris de Root Me

**La première limite**, c'est que l'on peut regretter qu'il n'y ait pas plus de directives pour inciter, sinon contraindre, à adopter de bonnes pratiques professionnelles.

Je ne suis pas encore renseigné sur les attentes dans les métiers qui emploient des professionnels du forensics, mais je me doute bien que savoir trouver l'épingle dans la botte de foin ne doit être que l'une des compétences requises. Tout comme du médecin légiste, il me semble que l'on

attendra de l'informaticien qu'il ne mette jamais son ADN partout avec ses gros doigts, et qu'il soit toujours en capacité de le démontrer. Car si la finalité est de produire une preuve, l'intégrité de cette dernière ne doit-elle pas, par définition, être incontestable ? Partant, une preuve ne se réduit pas à un fait ; elle tient sa nature du processus qui l'a révélée.

Aussi me semble-t-il qu'en plus de savoir produire une preuve, un professionnel du forensics doit être en mesure de prouver que cette preuve en est une, ce qui mobilise d'autres compétences, comme celle de savoir documenter ses investigations. Si l'on se contente de résoudre des challenges, il me semble que l'on risque trop de s'enfermer dans la logique de compétiteur de CTF, et de développer alors non seulement une conception, mais aussi des pratiques éloignées de ce que peut être le véritable forensics – en témoigne un script comme celui produit par TeamCTF-PRIME, qu'on pourrait d'emblée lancer sans se poser d'intéressantes questions.

Il faut sérieusement tempérer cette critique, car il serait parfaitement injuste d'exiger de Root Me qu'il joue un rôle qui n'est pas le sien. Rien que dans son nom, Root Me se présente clairement comme un site de challenges. Il n'a pas vocation à se substituer à des formations, mais à donner envie d'en poursuivre, voire à les compléter – sur ce point, j'espère que tous les enseignants et autres formateurs en sécurité informatique pressent leurs étudiants de s'y inscrire.

Par ailleurs, j'ai déjà dit qu'il est possible de proposer une solution à un challenge pour qu'elle soit publiée. Or en rédiger une implique de se livrer à un exercice qui constitue la base de l'écriture d'un rapport. On ne peut donc pas dire que Root Me ne fait que valoriser une pratique secrète du hacking, et c'est tant mieux.

**Une deuxième limite** est plus spécifique aux challenges de forensics. C'est bien beau de prétendre qu'il est interdit de résoudre un challenge en procédant par pifomètre — deviner un mot de passe —, encore faudrait-il que ce ne soit pas le seul moyen possible de parvenir à ce beau résultat. Or, force est de constater que ce n'est pas toujours le cas, que la solution, ou du moins une partie de cette dernière, ne s'obtient pas toujours en mobilisant la seule la technique, mais le flair, le nez planté dans les indices. Ici encore, c'est une critique qu'il faut tempérer. Dans la réalité, qui est un labyrinthe, il est bien nécessaire de faire des choix qui sont arbitraires, et qui sont d'ailleurs inconsciemment guidés par l'expérience, c'est-à-dire de la compétence si parfaitement intégrée qu'on n'est même plus conscient qu'on la mobilise. Par ailleurs, il faut savoir se montrer efficace, et le puriste peut perdre beaucoup de temps pour parvenir à un résultat.

D'ailleurs, cela se lit dans certaines solutions, où les auteurs présentent des programmes qui avalent le dump et recrachent le flag. S'il est bien entendu que c'est une manière de documenter le travail – *the code is the doc*, n'est-ce pas ? –, et qu'il est parfois nécessaire de coder, il me semble que c'est un peu perdre du temps pour rien, le premier impératif m'apparaissant de faire vite et bien. Après, il est parfaitement possible que les auteurs de tels programmes les aient conçus très rapidement… À voir.

Dans le même registre critique, l'on peut regretter qu'un challenge forensics repose intégralement ou en partie sur le principe que je qualifierai de « la lettre volée ». Je fais référence à la célèbre nouvelle d'Edgar Allan Poe, dont je ne citerai que ce passage pour ne pas en ruiner la lecture – d'ailleurs incontournable pour qui s'intéresse au forensics, il me semble :

"*Perhaps it is the very simplicity of the thing which puts you at fault,*" *said my friend.*

"*What nonsense you do talk!*" *replied the Prefect, laughing heartily.*

"*Perhaps the mystery is a little too plain,*" *said Dupin.*

"*Oh, good heavens! who ever heard of such an idea?*"

"*A little too self-evident.*"

"*Ha! ha! ha! -- ha! ha! ha! -- ho! ho! ho!*" *roared our visitor, profoundly amused,* "*oh, Dupin, you will be the death of me yet!*"

Ce type de challenge est particulièrement exaspérant, mais il est vrai que la critique doit être encore une fois tempérée, car ce vice est aussi sa vertu. En effet, un tel challenge est certainement des plus formateurs, quand l'on constate rétrospectivement — importance de relire les notes que j'ai évoquées — la quantité de techniques qu'on a mobilisée, et l'effort qu'on y investit… pour pas grand-chose. Si l'apprentissage par renforcement négatif à la Skinner n'est pas une blague, il est à parier que certains se souviendront longtemps de ce qu'ils ont appris lors d'un tel challenge, en premier lieu de faire le tour des évidences — ce qui, de manière assez intéressante, est le terme anglais pour désigner une preuve — avant de chercher ailleurs.

**Une troisième et dernière limite** est qu'aucun challenge de forensics n'exige de démontrer que quelque chose n'existe pas. On connaît la formule, reprise par Donald Rumsfeld, alors Secretary of Defense, selon laquelle l'absence de preuve n'est pas la preuve de l'absence, et le dessein parfaitement douteux dans lequel elle a été employée. C'est justement pour éviter de forger des esprits aussi tordus qu'il serait intéressant que des challenges de forensics consistent à démontrer que l'on ne peut trouver quelque chose.

L'enjeu peut être explicite : il s'agira de démontrer qu'un système n'était pas infecté, comme le médecin légiste démontrera que la victime n'a pas été empoisonnée. Il peut être aussi implicite, car permettre à celui qui nous a missionnés de savoir qu'un fait ne s'est pas produit peut grandement influencer les enseignements qu'il en tirera. On imagine un juge, qui n'inculpera pas pareillement son témoin assisté s'il sait que ce dernier non seulement a infecté une machine, mais qu'il a de plus exfiltré des données.

Certes, une fois encore, c'est sans doute attendre de Root Me ce qu'il n'a pas vocation à apporter. Toutefois, ce n'est qu'en partie vrai, et je pense qu'il faudrait que les auteurs de challenges de forensics en élaborent dont la finalité serait de prouver une absence. À quoi pourrait ressembler le flag dans ses conditions ? Je ne le sais pas. À ces génies de trouver. Ils ont largement démontré qu'ils savaient enterrer un flag ; il n'appartient qu'à eux de passer le cap suivant, qui consiste à le faire disparaître. Nom du challenge : 64rc1m0r3.

## Bref, surtout ne pas se priver de Root Me

Root Me est passionnant, j'espère l'avoir bien dit. Pédagogiquement, on est dans un registre qui n'a pas été sans me renvoyer à un cursus de MBA, où le principe très anglo-saxon était que tout l'enseignement se fondait sur des études de cas. Il a ses limites, car il est parfaitement possible de

ne s'en tenir qu'au cas. Toutefois, il n'appartient qu'à l'étudiant de repousser les limites en profitant de l'opportunité unique qui lui est offerte de s'intéresser à un sujet, contre des points, pour l'approfondir au-delà de ses seuls besoins de l'instant.

Sur Root Me, du moins pour ce qui concerne les challenges forensics que je rappelle être les seuls que j'ai relevés – pour l'instant, évidemment ! –, la difficulté m'est apparue très bien dosée. J'en reste même étonné, tant il me semble rétrospectivement avoir été pris par la main pour rentrer dans le forensics, tout particulièrement pour apprendre à utiliser l'incontournable volatility.

Un grand merci donc à ceux qui ont créé et animent Root Me, ainsi qu'à ceux qui ont produit les challenges. Ceux-là, je les ai souvent voués aux gémonies ces derniers temps pour m'être jeté dans une telle Géhenne, mais c'est bien tout le jeu : Thanat0s, sambecks, eilco, N1lux, makhno, sourcePerrier, Bernstein, fraf, Futex, cryptax, koma, Siras, franb, dans le désordre et en espérant n'avoir oublié personne.

Je ne saurai donc trop chaudement recommander à vous tous, petits et grands, d'aller vous amuser sur Root Me. Et comme il existe certainement de nombreux sites du même genre, j'invite ceux qui les ont testés à se manifester auprès de notre rédacteur en chef préféré pour lui proposer d'en parler dans ces colonnes, ou alors de s'inscrire à un Meetup du fort sympathique chapitre France d'OWASP (https://www.meetup.com/fr-FR/owasp-france/) pour en faire l'objet d'un flashpoint, voire d'un atelier.

## Les rootmistes existent, j'en ai rencontré

À l'occasion d'un Meetup automnal du chapitre France d'OWASP — on en profite pour remercier les gentils organisateurs, car c'est du boulot, et il est bien fait —, j'ai invité qui voulait à témoigner de son expérience de Root Me a se manifester. Bruts de décoffrage, voici quelques témoignages.

### Henry, étudiant à Epitech

*J'ai découvert Root Me lorsque j'étais en première année dans mon école Epitech. Il y avait une association de cyber sécurité qui proposait une « initiation » à ce domaine, en commençant par Root Me.*

*Lorsque j'ai commencé Root Me, je n'avais pas assez d'intérêt pour m'y plonger. C'est lorsque j'ai commencé à effectuer des CTF que cet intérêt est revenu. Le côté jeu et compétition était devenu une bonne source de motivation.*

*J'ai commencé Root Me avec les challenges web. Par la suite, j'ai effectué des challenges de toutes les catégories pour me diversifier. Récemment, je suis revenu sur la catégorie forensic pour faire les challenges « command & control - niveau 3 et 4 ». J'ai également découvert un nouveau challenge de stéganographie (« points jaunes »), très fun et facile à faire, je le recommande à tous !*

*Effectuer des challenges sur Root Me en parallèle de mon cursus scolaire m'a permis de mieux comprendre certains concepts de sécurité, mais aussi d'adopter des bonnes pratiques dans mes projets d'école et lors de mes stages. Durant mon année scolaire à l'étranger, je me suis basé sur un challenge pour prouver que les captchas peuvent être contournés avec un simple script Python.*

*Le point fort de Root Me est de fournir des challenges variés et ludiques qui nous permettent d'apprendre beaucoup sur la sécurité informatique.*

*Pour moi, Root Me est une plateforme idéale pour monter en compétences dans la cyber sécurité, surtout pour des étudiants comme moi qui n'ont pas de vraie formation dans le domaine. Cependant, je pense que des articles sur les CVE récentes pourraient être intéressants pour concrétiser encore plus l'apprentissage.*

### T0t0r0, analyste en cybersécurité dans une ESN

*J'ai découvert « Root Me »/RM et les CTF vers la fin de mes études, par curiosité intellectuelle, en faisant du web, réseau, script, stéga, crypto, de manière assez sporadique et basique.*

*Je m'y suis remis plus sérieusement fin 2018 grâce à la compétition et l'émulation au sein d'« Hackademint » (hackademint.org et leur Discord pour les curieux) et à la découverte de la communauté française infosec très présente sur Twitter. C'est devenu une drogue et une passion pendant plusieurs mois, ne comptant plus les heures passées sur cette plateforme.*

*Je me suis notamment acharné sur les challenges de web-serveur, les plus « simples » , intéressants, compréhensibles, pratiques et facilement reproductibles pour moi. J'ai aussi découvert une catégorie que j'apprécie toujours autant depuis : le forensic. D'ici la fin de l'année je compte commencer les aspects plus « bas niveau ».*

*Il y a des points très exclusifs que j'apprécie sur RM :*

- *La communauté très diverse, leur gentillesse, leur technicité. Je ne peux déjà pas exprimer la joie que j'éprouve en résolvant certains challenges, mais encore moins celle lorsque je lis et tente de reproduire ensuite les solutions des autres membres de la communauté, ce qui pousse d'ailleurs à partager ses propres solutions.*
- *La diversité et la qualité des challenges : RM est vaste, très qualitatif et assez complet. Étant fan de challenges, c'est une plateforme ludique incontournable pour moi : le temps passe vite, on apprend énormément en résolvant des challenges et à travers nos propres recherches, fructueuses ou non, on découvre de nombreux détails et aspects jusqu'alors inconnus, parfois utiles au travail comme à la maison.*

## Références utiles

L'outil le plus utilisé dans ces challenges est l'incontournable volatility, écrit en Python. Son source peut être téléchargé sur GitHub :

https://github.com/volatilityfoundation

Alternativement, il est possible d'en télécharger un binaire :

https://www.volatilityfoundation.org/26

Pour se former à volatility, il est bon de lire *The Art of Memory Analysis*, publié par John Wiley & Sons (2014), et le blog *Volatility Labs* :

https://volatility-labs.blogspot.com/

Les autres outils les plus communs sont ceux des suites Sysinternals et Skeuth Tools, ainsi que TRiD. Pour le reste, il paraît assez inévitable d'apprendre à programmer en Python pour réaliser rapidement des outils ad hoc.

Ceux qui travaillent sous Windows peuvent faire tourner Kali Linux dans une VM à l'aide de Hyper-V, cela fonctionne parfaitement.

Enfin, le texte intégral de La lettre volée d'Edgar Allan Poe est accessible ici :

https://americanliterature.com/author/edgar-allan-poe/short-story/the-purloined-letter

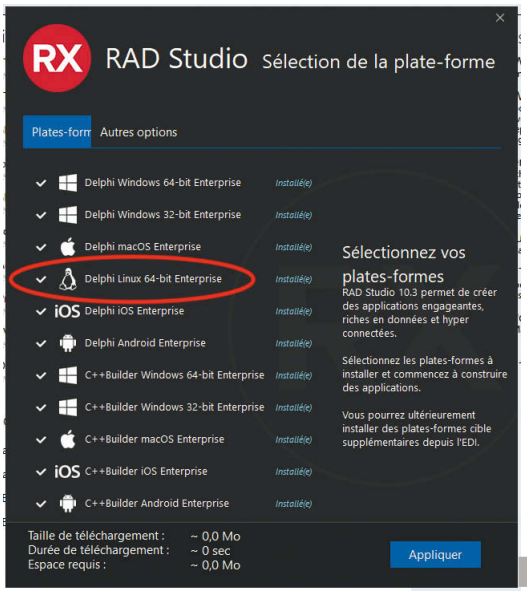
**Patrick Prémartin**
*MVP Embarcadero (Delphi), prestataire informatique freelance, auteur, formateur et occasionnellement conférencier. Utilisateur du Pascal depuis 1990 et de Delphi depuis sa sortie en 1995.*

# Delphi revient sur Linux

*À vrai dire Delphi n'est jamais totalement parti de Linux, mais cette fois-ci on n'a plus aucune excuse pour ne pas y aller aussi !*

**Avertissement : cet article ne contient aucun spoiler ni remarque sur Kylix en dehors de ce paragraphe.**

Comme je l'écrivais dans Programmez ! 217 d'avril 2018, Delphi a su évoluer au fil des années et propose régulièrement des nouveautés intéressantes pour les développeurs.
Cela fait plusieurs années qu'Embarcadero propose un compilateur Linux (64 bits pour processeurs Intel) aux utilisateurs des licences Entreprise et Architecte. Jusque-là il n'y avait pas de version de FireMonkey (le framework de composants visuels multi-plates-formes de Delphi) fournie en standard ni de concepteur de fiche. On pouvait bien entendu faire des applications graphiques, mais en utilisant les API des environnements de bureau où nous cantonner à des logiciels en console ou serveur. La demande d'intégrer des outils visuels pour faire des développements en mode desktop était forte parmi les utilisateurs de RAD Studio et Delphi. Il existait un produit payant pour contourner le problème : FMX Linux.
FMX Linux est une extension de FireMonkey. Il permet d'ajouter Linux aux cibles de compilation de tout projet FMX et ajoute les librairies nécessaires pour tourner correctement avec l'environnement de bureau d'Ubuntu (et d'autres distributions non officiellement supportées).
Entre des évolutions de syntaxe du Pascal en 10.3.0, le compilateur macOS 64 bits en 10.3.2 et la sortie du compilateur Android 64 bits dans la 10.3.3 l'année 2019 a été plutôt riche en évolutions. Embarcadero a également signé un contrat de distribution auprès de l'éditeur de FMX Linux en juin et nous en fait désormais bénéficier gratuitement.
Tout détenteur d'une licence à jour des éditions Entreprise ou Architecte de RAD Studio et Delphi avaient le compilateur Linux intégré pour ses logiciels « serveur ». Ils peuvent désormais utiliser FMX Linux sans coût supplémentaire.
Les licences Community Edition (gratuite pour un usage personnel) et Professional ne proposent pas le compilateur Linux et ne sont donc pas concernées par la suite de cet article (sauf la façon de développer qui reste commune).
Certaines licences Academic (pour les étudiants et enseignants) incluent le compilateur Linux, d'autres pas. Renseignez-vous auprès du distributeur Barnsten pour les détails.
Si vous n'avez pas de licence adaptée, vous pouvez utiliser une version d'évaluation avec sa licence de 30 jours disponible depuis le site d'Embarcadero afin de pratiquer un peu et vous faire la main.
Contrairement à d'autres outils de développement en Pascal permettant de travailler sous Linux, Delphi et RAD Studio restent des outils de développement sous Windows, mais intègrent les fonctionnalités nécessaires pour créer des logiciels et applications compilées nativement pour iOS (ARM 32/64 bits), Android (ARM 32/64 bits), macOS (Intel, 32/64 bits), Windows (Intel/AMD&co, 32/64 bits) et Linux (Intel/AMD&co, 64 bits).
Vous l'aurez compris avec cette longue introduction : je vais vous montrer comment faire du développement Linux depuis Windows avec Delphi.



1

Pour la suite il vous faut donc un PC sous Windows et une machine sous Ubuntu (ou équivalent). Vous pourrez bien entendu travailler depuis un ordinateur sous Linux avec une machine virtuelle Windows ou tout depuis Windows en activant le sous-système Linux que Microsoft embarque désormais dans Windows 10.

## INSTALLATION

### Étape 1 : sous Windows

Lors de l'installation de Delphi ou RAD Studio vous devez choisir les compilateurs qui vous intéressent. Pour travailler sous Linux, il faut cocher la case « Delphi Linux 64-bit ». 1
Si vous aviez sauté cette étape, rendez-vous dans l'option « Outils / Gérer les plates-formes » de l'IDE. Elle affiche le même écran et permet de rattraper l'oubli.
FMX Linux est distribué sur le magasin d'applications et de composants d'Embarcadero. On y accède par l'option « Outils / Gestionnaire de packages GetIt ». Vous pouvez ensuite chercher « FMX Linux » ou vous rendre dans la rubrique « composants ».
L'installation de FMX Linux se fait en deux parties : la librairie et ses exemples. 2
Cliquez sur la ligne du paquet à installer, sur le bouton INSTALLER et suivez les indications.

### Étape 2 : sous Linux

L'installation de Delphi / RAD Studio ajoute un certain nombre de programmes complémentaires dans l'arborescence du produit.

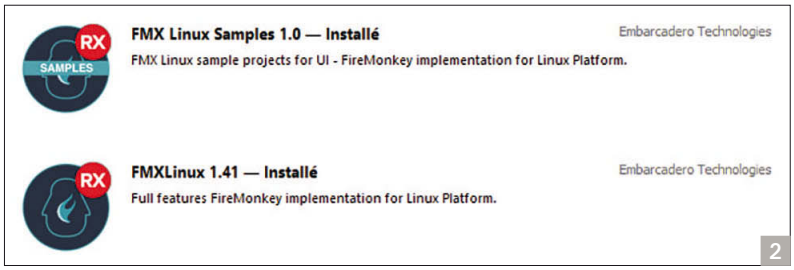
C'est dans le dossier « c:\Programmes (86)\Embarcadero\Studio\20.0\PAServer » que vous trouverez le fichier LinuxPAServer20.0.tar.gz à installer sur Linux. Il servira de passerelle entre Linux et l'environnement de développement sous Windows.
Une fois le fichier copié sur votre machine sous Linux, un simple « tar -xvf LinuxPAServer20.0.tar.gz » le décompresse pour donner le dossier « PAServer-20.0 » où se trouve le programme « paserver ».

```
pprem@ubuntu:~$ tar -xvf LinuxPAServer20.0.tar.gz
PAServer-20.0/
PAServer-20.0/linuxgdb
PAServer-20.0/paconsole
PAServer-20.0/paserver
PAServer-20.0/paserver.config
pprem@ubuntu:~$ ls
Desktop    examples.desktop          PAServer-20.0  Templates
Documents  LinuxPAServer20.0.tar.gz  Pictures       Videos
Downloads  Music                     Public
pprem@ubuntu:~$
```

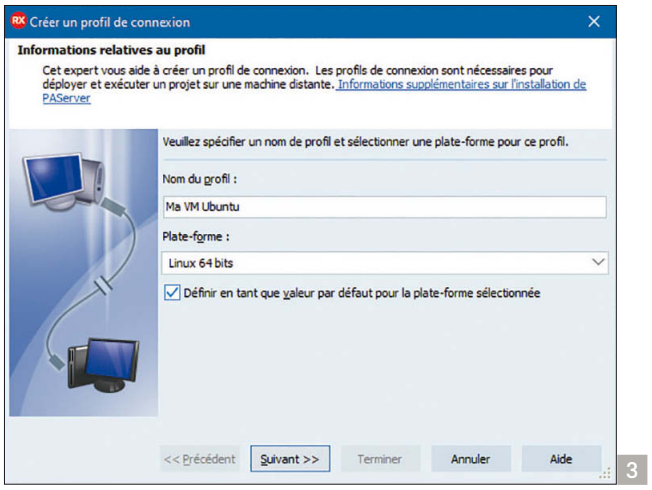
Selon votre distribution il y aura peut-être des paquets additionnels à installer, dont le SDK de la distribution et GCC pour C++. Un tuto de Craig Chapman est disponible sur son blog pour Ubuntu et Red Hat :
https://chapmanworld.com/2016/12/29/configure-delphi-and-redhat-or-ubuntu-for-linux-development/
Exécutez « PAServer-20.0/paserver » depuis une fenêtre du terminal, tapez les commandes « v » pour passer en mode verbeux, « i » pour avoir les infos réseau utiles, notez votre IP puis retournez sous Delphi pour terminer la configuration.

## Étape 3 : dans Delphi / RAD Studio

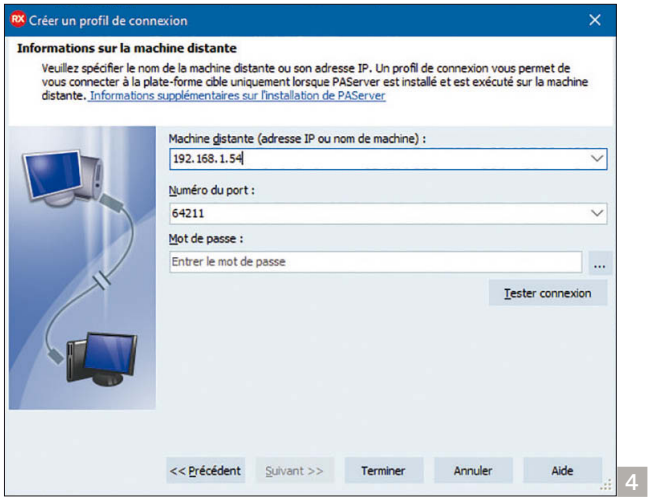
Pour terminer la configuration, il faut indiquer à Delphi quel SDK il doit utiliser et avec qui il doit travailler. Deux dernières étapes avant de pouvoir faire notre premier programme.
Allez dans « Outils / Options » puis dans « Déploiement / Gestionnaire de profils de connexion ».
Ajoutez un nouveau profil en choisissant Linux dans la liste. 3
Indiquez l'IP de la machine Linux, modifiez le port si vous désirez (pensez à le changer aussi côté Linux en paramètre de ligne de commande du PAServer) et un mot de passe si vous ne voulez pas que d'autres puissent y accéder. Mot de passe que vous aurez au préalable saisi sur PAServer lorsque vous l'avez lancé côté Linux.
En local pas besoin de mot de passe, mais si vous installez PAServer sur un serveur en ligne, pour déployer ou déboguer à distance, c'est plus que fortement recommandé d'en mettre un et de le changer régulièrement. 4
Enregistrez la fenêtre de dialogue pour que l'IDE prenne en compte cette configuration.
Dernière étape : le SDK Linux à utiliser.
Revenez dans « Outils / Options » puis « Déploiement / Gestionnaire de SDK ». Ajoutez un SDK Linux et laissez l'environnement récupérer les fichiers dont il a besoin. 5
Enregistrez, c'est plié.
Ces étapes d'installation sont à faire à chaque version majeure de Delphi. Les mises à jour intermédiaires peuvent conserver ces paramètres, mais vous aurez probablement à mettre à jour le PAServer, car chaque version de Delphi vient avec une nouvelle version de cet utilitaire.
De plus si vous changez de version de Linux ou mettez votre distribution à jour, pensez à rafraîchir les fichiers de son SDK dans



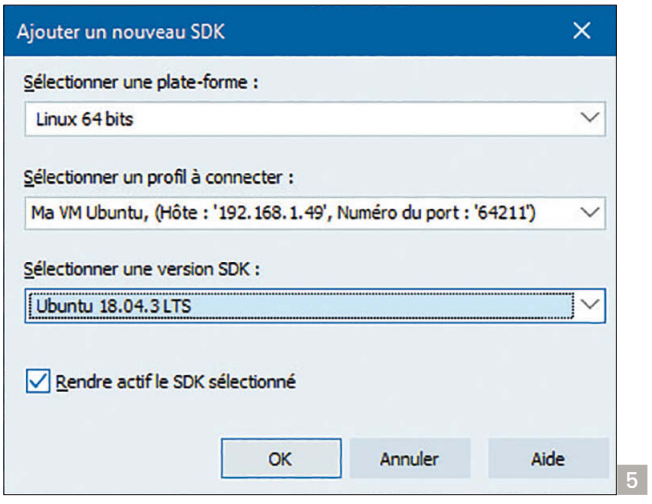
2



3



4



5

Delphi sinon le lieur est susceptible d'avoir un problème lors de la compilation.

## L'utilisation

Je vous propose de faire plusieurs petits programmes simples pour voir que tout fonctionne et surtout vous donner une idée des possibilités.

Dans Delphi, la plupart des librairies non graphiques fournies tournent sur toutes les plates-formes ciblées. C'est globalement aussi le cas des composants non visuels. Pour les composants visuels en revanche il faut partir sur l'arborescence FireMonkey en travaillant sur des projets multi-périphériques.

Commençons par la base pour tester votre environnement.

### « Hello World » en console

Créez une nouvelle application en choisissant « Application console » depuis « Fichier / Nouveau / Autre / Projets Delphi ». L'assistant de création ouvre un programme standard Pascal. Vous allez juste ajouter une ligne de code dans le bloc d'exécution. Le programme ressemble à ça :

```
program Project1;
{$APPTYPE CONSOLE}
{$R *.res}
uses
 System.SysUtils;
begin
 try
  writeln('Hello World');
 except
  on E: Exception do
   writeln(E.ClassName, ': ', E.Message);
 end;
end.
```

Si vous faites F9, ça va s'exécuter sous Windows en ouvrant une fenêtre de commande qui sera fermée juste après puisque c'est un simple affichage sans boucle d'attente.

Pour l'exécuter sous Linux, il faut d'abord ajouter la plate-forme au projet depuis le gestionnaire de programmes. L'option se trouve dans le menu contextuel des plates-formes cibles. 6 7

En laissant Linux actif par défaut il suffit de refaire F9 pour que Delphi envoie le programme compilé à la machine Linux par défaut sur laquelle s'exécute PAServer. 8

Si vous obtenez des erreurs de linkage ou de compilation, c'est qu'il y a un souci côté Linux, donc vérifiez quels paquets de développement il vous manque, installez-les, mettez à jour le SDK dans Delphi et recompilez.

### « Hello World » en mode fenêtré

Cette fois-ci, partons sur une « vraie » application, avec un bouton et une boîte de dialogue.

Créez un nouveau projet en choisissant « application multi-périphérique » et prenez le modèle vierge depuis l'assistant de création.

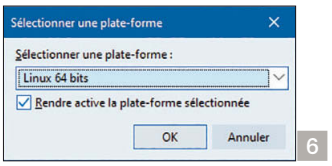
Sur la fiche qui apparaît, ajoutez un bouton dont vous mettrez la propriété Align à Center.

Double cliquez sur le bouton ou dans ses événements sur onClick pour ajouter une action lors du clic sur le bouton. Dans la procédure générée, faites l'affichage d'un texte sous forme de boîte d'alerte. Voici le code correspondant :

```
procedure TForm1.Button1Click(Sender: TObject);
begin
 ShowMessage('Hello World');
end;
```

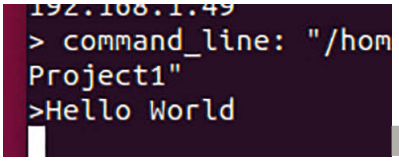
Exécutez le programme par la touche F9 sous Windows. 9

Et maintenant, ajoutez notre cible du jour par le menu contextuel sur le nom du projet dans le gestionnaire de projet puis l'option « Add Linux Platform », compilez et exécutez le programme.

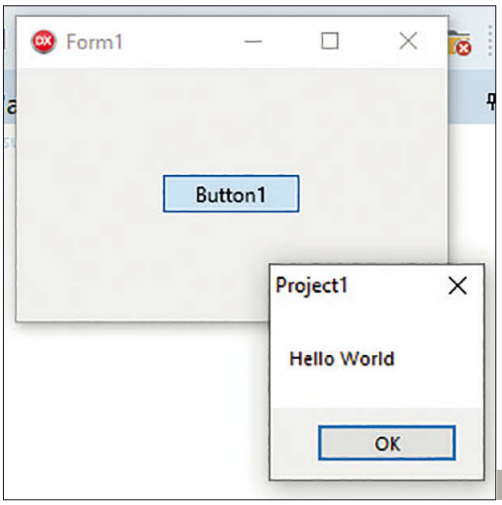


6

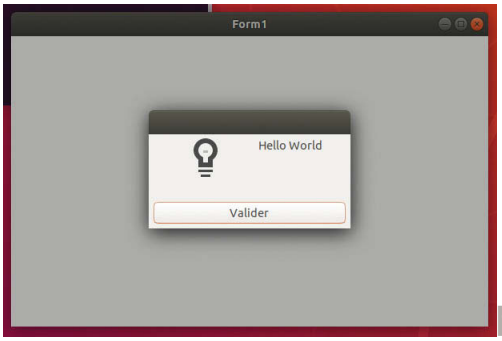
7



8



9



10

Attention : il ne faut pas faire comme pour l'ajout de la cible « Linux » à un projet en console. Ça ne suffirait pas. FMX Linux s'ajoute en direct depuis le projet. 10
Ce n'est pas plus compliqué de créer une application Linux depuis Delphi.

## Voir plus loin

Comme pour tous les systèmes d'exploitation gérés nous avons accès aux SDK et librairies du système d'exploitation. Nos appels vers des ressources externes en Pascal se font à peu près de la même façon que pour les DLL Windows.
Les fonctions standards de l'OS sont aussi accessibles, par exemple la fonction fork() qui ne sert que sous Linux et qui est disponible dans les unités correspondantes.
Les librairies open source qui sont la plupart du temps en C sont aussi mélangeables avec le code Pascal quand on sait s'y prendre (ce n'est pas très compliqué et il y a des outils pour générer les fichiers d'interfaces).
Bien entendu on peut conditionner des blocs de code dans nos sources pour viser une plate-forme plutôt qu'une autre. Ces directives de compilation sont très utilisées dans les projets open source en Pascal et dans les unités Delphi qui gèrent des versions différentes des compilateurs, des gestionnaires de mémoire (comme ARC sur mobile) et des systèmes d'exploitation.
Voici comment faire :

```
{$IF Defined(ANDROID)}
// Code ANDROID
{$ELSEIF Defined(IOS)}
// Code IOS
{$ELSEIF Defined(MACOS)}
// Code macOS
{$ELSEIF Defined(MSWINDOWS)}
// Code MS Windows
{$ELSEIF Defined(LINUX)}
// Code Linux
{$ENDIF}
```

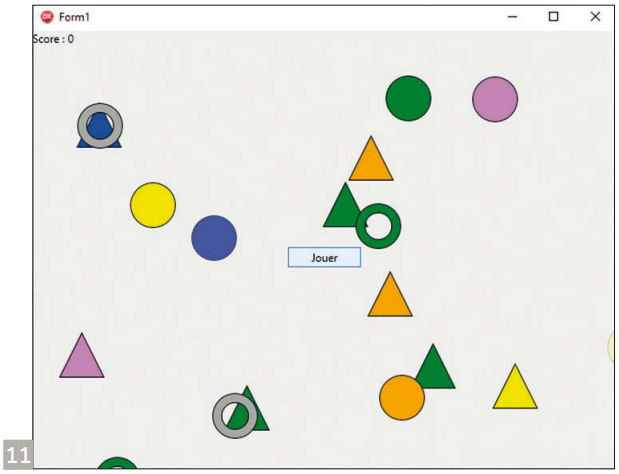
Toujours tester macOS après iOS, car il est aussi valide dans le cas d'iOS et s'exécuterait à sa place.
Les routines d'accès aux fichiers sont elles aussi multiplateforme en passant par l'unité System.IOUtils et les méthodes des classes TPath (pour gérer les arborescences), TFile (pour l'accès physique aux fichiers) et TDirectory (pour l'accès physique aux dossiers). N'hésitez pas à basculer dessus dans vos projets existants même en VCL.
Les frameworks comme FireDAC sont fonctionnels en desktop comme en mobile, avec certaines restrictions sur les moteurs de bases de données disponibles.
J'utilise généralement Interbase, Firebird ou SQLite pour les bases de données locales et des accès à des API depuis les applications mobiles pour les bases de données distantes. Il y a aussi d'autres solutions avec ou sans licence complémentaire (DataSnap, Web Broker, RAD Server, …).
On reste dans un logiciel de développement puissant et ouvert sur l'extérieur avec une vraie capacité à ne développer qu'une fois et compiler sur plusieurs environnements différents.
Dans les usages du compilateur Linux on peut aussi faire nos propres serveurs d'API ou Web avec Web Broker, des composants Indy ou à notre sauce. Ça peut être intéressant quand on n'a pas envie d'avoir des serveurs sous Windows ou qu'on ne veut pas diffuser les sources de nos produits. On compile le code d'un programme console qui sera utilisé en serveur autonome, en CGI ou en extension à Apache.
Le compilateur Linux fourni est en 64 bits pour des processeurs à architecture Intel. Pas d'ARM pour le moment, mais ça peut changer. Les Raspberry Pi sont assez régulièrement cités quand on parle d'évolution du produit et il n'est pas exclu de voir un compilateur Linux ARM pour cibler cette plate-forme. En attendant, on peut toujours y utiliser une distribution Android.

## Finir en beauté

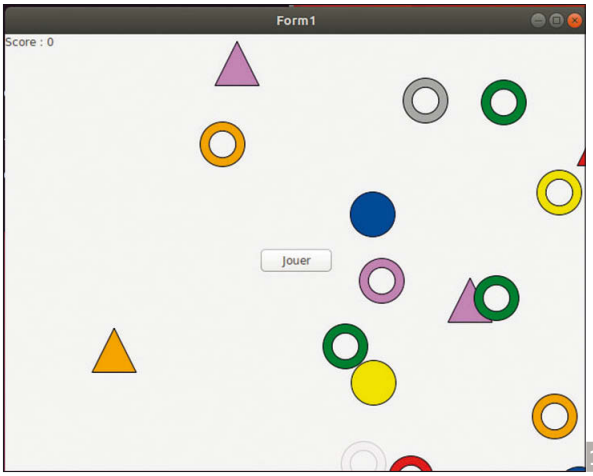
Pour conclure cette présentation des capacités Linux de Delphi et comme c'est un peu le truc pour lequel nous sommes nombreux à avoir plongé dans le codage, je vous propose de faire un petit jeu qui fonctionnera sur toutes les plates-formes y compris Linux puisqu'il suffira de compiler sur la bonne cible.
L'idée est simple : on affiche des formes géométriques à l'écran et l'utilisateur ne doit cliquer que sur les vertes pour gagner des points.
S'il clique sur autre chose, il perd des points.
S'il tombe à zéro il perd.
Pour ça il faut créer un nouveau projet multi-plate-forme.
Dans la fiche générée on met :

- 1 TButton (name = btnJouer, align = Center, Text = Jouer) qui déclenchera une partie
- 1 TTimer (enabled = false, interval = 300) qui créera les formes sur lesquelles cliquer
- 1 TLabel (name = lblScore, align = Top) pour afficher le score

Dans le code source de la classe TForm on ajoute en public les propriétés Score (integer) et JeuOnOff (boolean).
Sur la fiche on active l'événement onCreate dans lequel on met le score à 0 et le mode de jeu à False.
Dans le setter de la propriété Score on met à jour le texte de lblScore avec le texte « Score : XXX » (où XXX sera remplacé par le nombre de points).
Dans le setter de la propriété JeuOnOff on rend visible le bouton btnJouer si la valeur est à false et on met à 0 le score dans le cas du démarrage d'une nouvelle partie (donc quand on passe à true et qu'on était à false).
On change aussi la propriété HitTest de toutes les formes présentes à l'écran pour désactiver le onClick lorsqu'on n'est pas en cours de jeu et la réactiver dans le cas contraire.
Pour les formes, nous allons passer par la librairie Radiant Shapes disponibles sur GetIt. Elle fournit un grand nombre de formes vectorielles sous forme de composants visuels très pratiques.
Pour l'installer, vous suffit d'aller dans « Outils / Gestionnaire de packages GetIt » et d'y chercher « Radiant Shapes » ou « Bonus ».
Enregistrez votre projet avant d'y aller. Après l'installation Delphi devrait se relancer pour prendre en compte ces nouveaux composants.
Rouvrez votre projet et ajoutez les unités Radiant.Shapes et FMX.Ani aux unités utilisées en implémentation dans le code.
Sur l'événement onInterval du timer, on ajoute des variables locales



11



12

pour un TRadiantTriangle, un TRadiantCircle, un TRadiantRing et un TFloatAnimation.

Dans le code de l'événement, on crée aléatoirement l'une des trois formes et on lui attache une animation touchant à son opacité. Cette animation changera d'effet dans la durée. On joue sur son événement onFinish pour cela.

On attache au onClick de chaque forme une méthode clickOK ou clickKO selon sa couleur. Seules les formes vertes auront clickOK dont le rôle est d'augmenter le score, alors que l'autre est de le baisser.

Lors de la fin de l'animation d'une forme ou d'un clic dessus on supprime son instance ce qui supprime aussi les composants dépendants donc dans notre cas leur animation.

Sur l'événement onShow de la fiche, on s'assure que le timer est activé et on le désactive sur l'événement onHide. Inutile qu'il anime l'écran pendant qu'on ne le voit pas.

Je vous laisse télécharger et consulter les sources du projet. Vous devriez voir la simplicité de codage et comprendre sans problème la mécanique du jeu facilitée par la programmation événementielle et les actions dans les différentes propriétés.

On peut exécuter ce projet sous Windows pour le tester, puis l'envoyer sur un smartphone ou une tablette (iOS ou Android), un Mac et enfin un Linux. Les opérations de débogage sont possibles sur toutes les plates-formes avec les mêmes possibilités (points d'arrêts, surveillance et modification de zones mémoires …).

Au moment d'écrire cet article, la librairie Radiant Shapes ne fournissait pas encore les DCU pour Linux (des fichiers précompilés liés à Delphi), mais comme on en a les sources il suffit de les ajouter au projet pour que la compilation passe automatiquement. Depuis le gestionnaire de projet ajoutez simplement le fichier « C:\Program Files (x86)\Raize\RadiantShapes\Source\Radiant.Shapes.pas ». 11 12

Ce jeu est très sommaire. C'est le même principe qui m'a servi de base pour Pumpkin Killer que vous pouvez télécharger sur mobile depuis le Play Store et l'App Store ou le jeu de Noël dont j'ai fait un pas à pas en vidéo (dispo sur mon blog) lors du CodeRage 2018.

Delphi permet de faire des jeux plus évolués comme bien entendu des logiciels « plus sérieux » comme des gestions de bases de données ou des serveurs, même si dans ce cas il faut coder la couche métier.

Embarcadero fournit un grand nombre de projets exemples pour FireMonkey. Ils sont pour la plupart utilisables aussi sous Linux. N'hésitez pas à vous y reporter.

Bon code !

**Lilian Saget-Lethias**
*Software Architect*
**Paris Deno Creator**
@lsagetlethias

# Deno
# Première approche

***Avant de commencer, il est très important qu'au moment où cet article est écrit, Deno est toujours en cours de développement. Donc, tout code produit doit être considéré comme instable, du fait des changements potentiels, et non anticipés, de l'API. Nous nous baserons pour la suite sur la version 0.21.0. Enfin, il faut aussi préciser que Deno n'a pas pour but de remplacer Node ou de fusionner avec.***

## INTRODUCTION & ARCHITECTURE

Deno est un *runtime cross-platform*, c'est-à-dire un environnement d'exécution, basé sur le moteur V8 (https://v8.dev/) de Google, développé avec le langage Rust (https://www.rust-lang.org/), et construit avec Tokio (https://github.com/tokio-rs/tokio) pour le système d'*event-loop*.

### Le problème de Node

Deno a été présenté par son créateur, **Ryan Dahl (@ry)**, lors de la JSConf européenne en juin 2018, à peine 1 mois après les premiers commits.
Lors de cette présentation, Dahl exposa dix défauts d'architecture de Node (auxquels il s'attribue la faute). En résumé:

- Node.js a évolué avec les *callbacks* au détriment des Promesses qui étaient pourtant présentent dans les premières versions du V8
- La sécurité du contexte applicatif
- GYP (*Generate Your Projects*), le système de compilation forçant entre autres les utilisateurs à écrire leurs *bindings* (liaisons entre Node et V8) en C++ alors que V8 ne l'utilise plus.
- Le gestionnaire de paquets de dépendances, NPM, intrinsèquement lié au système de require de Node. Les modules NPM sont stockés, jusqu'à maintenant sur un seul service centralisé et gérer par une entreprise privée. Enfin, le fichier package.json est devenu trop centré sur le projet plutôt que sur le code technique lui-même (licence, description, repository, etc.).
- Le dossier node_modules est devenu beaucoup trop lourd et complexe avec les années rendant compliquer l'algorithme de résolution des modules. Et surtout, l'utilisation des node_modules et du require est une divergence des standards établis par les navigateurs.
- La syntaxe de require omettant les extensions de fichier en ".*js*", qui, comme le dernier point, diffère du standard des navigateurs. De plus, l'algorithme de résolution de module est obligé de parcourir plusieurs dossiers et fichiers avant de trouver le module demandé.
- Le point d'entrée nommé index.js devenu inutile après que le require soit devenu capable de supporter le fichier package.json
- L'absence de l'objet window présent dans les navigateurs empêchant tout isomorphisme

Finalement, le point négatif global est que Node a, au fil du temps, dépriorisé le système de saturation d'événements d'entrée-sortie au profit de la résolution de problèmes du système de modules.

### Les solutions de Deno

Dahl se lança dans Deno avec pour but de résoudre la majeure partie des problèmes de Node. Pour y parvenir, la technologie se base sur un ensemble de règles et de paradigmes permettant aux futures évolutions de suivre la ligne directrice :

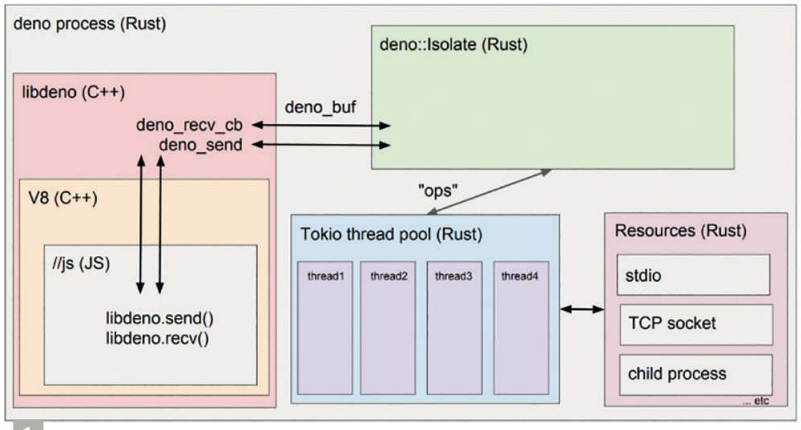
**Support natif de TypeScript**
Point important pour le créateur qui attache un intérêt très particulier pour le langage. Nous avons vu au fil des années Node se débattre avec le maintien du support des nouveautés de V8 et d'ECMAScript sans avoir à casser l'existant de l'API. C'est fini avec Deno. Il donne la possibilité d'utiliser TypeScript d'emblée sans configuration initiale de son application. L'utilisation est en revanche restreinte à la configuration native du compilateur par défaut. Un fichier tsconfig.json peut toutefois être donné au compilateur grâce au *flag* --config=<fichier>.

**Isomorphisme avec le monde Web en supportant la syntaxe des modules ECMAScript et en bannissant la fonction** require()
Comme indiqué précédemment, Node pâtit d'une résolution de dépendance inefficace ; Deno résout le problème en étant plus explicite, simple, et direct, tout en étant conforme aux standards.
(import * as log from "https://deno.land/std/log/mod.ts";)

**Le code distant est récupéré et mis en cache localement**
À l'instar des node_modules, les dépendances nécessaires au bon fonctionnement d'un projet sont téléchargées et récupérées localement. En revanche, elles ne seront pas stockées au niveau du projet, mais plutôt dans le dossier global de cache de Deno (~/.deno/src par défaut). Une même version d'une dépendance n'a donc pas besoin d'être re-téléchargée qu'importe le nombre de projets locaux la nécessitant. À noter que ce fonctionnement ressemble à yarn plug'n'play (https://next.yarnpkg.com/features/pnp).

**Des permissions spécifiques doivent être explicitement données par l'utilisateur final**
À une ère où la sécurité de toutes nos applications est de mise, Deno englobe l'exécutable produit dans un mode bac à sable où chaque opération sortant du contexte d'exécution doit être autorisée. Un accès réseau par exemple doit être accordé par un "yes" explicite de l'utilisateur dans la ligne de commande ou par le *flag* --allow-net.
Encore une fois, Deno souhaite se rapprocher des paradigmes Web : accès à la webcam par un site web, par exemple.

**Un livrable, un exécutable**
Dans un souci de distribution efficace, Deno propose son propre *bundler* (deno bundle) créant au moment de la livraison un seul et unique consommable (.js) et plus tard, un seul binaire exécutable (deno compile).




1

**D'autres fonctions…**

Deno vise aussi de toujours terminer le programme en cas d'erreurs non traitées ; d'avoir un code JavaScript généré compatible avec les navigateurs du marché ; de supporter les promesses au plus haut niveau de l'application (top-level await, supporté par V8, en cours de livraison du côté de TypeScript) ; d'être capable de servir *over-HTTP* à une vitesse efficiente (si ce n'est plus rapide que Node).

## Ce que ne cible pas (du tout) Deno

**L'utilisation d'un manifeste de type package.json**

Un manifeste de gestion de dépendances n'est pas nécessaire pour un code récupérant lui-même ses dépendances.

**L'utilisation d'un manager de paquets type npm**

Pour les mêmes raisons, npm (ou équivalent) n'est pas et ne doit pas être essentiel au développement d'une application Deno.

**Isomorphisme Deno / Node**

Même si les deux technologies utilisent le même langage, les *designs* ne sont pas les mêmes et ne permettent donc pas de code isomorphique.

## Le modèle architectural

### Rust

Rust est le langage utilisé pour englober le moteur V8. C'est lui qui en expose les fonctionnalités isolées à travers une API utilisable en JavaScript. Ce lien, ou *binding*, appelé libdeno, est livré en l'état indépendamment du reste de l'infrastructure de Deno grâce à un module Rust appelé deno-core (une *crate* ; https://crates.io/crates/deno) consommé par la ligne de commande, la deno-cli. Cette *crate* peut donc être utilisée dans votre propre application Rust si vous en faites le choix.

La deno-cli quant à elle, fait lien entre la *crate* core, le compilateur TypeScript (compilation à chaud et cache du code final), et Tokio (une librairie d'*event-loop*).

Voici pour résumer un schéma du processus d'exécution : 1

### Tokio

Cette librairie écrite en Rust donne au langage une notion de programmation asynchrone et de programmation orientée événement. Nativement, Rust ne supporte pas la gestion des boucles événementielles, et a, jusqu'en 2014, utilisé la librairie libuv pour effectuer ses opérations entrée-sortie de manière asynchrone et multi-plate-forme et donc pallier à ce manquement.

Il est à noter que Node utilise encore aujourd'hui libuv dans son processus englobant V8.

Tokio est donc devenu par la suite la librairie de référence pour tout *asynchronous event-driven programming* en Rust.

Du point de vue de Deno, Tokio s'occupe donc de paralléliser l'ensemble des entrées-sorties asynchrones effectuées par les *bindings* V8 exposés dans l'isolat deno-core (étant la *crate* standalone Rust)

### V8

Enfin, comme cité plusieurs fois, l'ensemble de l'architecture repose sur le moteur d'interprétation JavaScript. Il est régulièrement mis à jour pour, entre autres, suivre les besoins des dernières versions de TypeScript. À l'heure où cet article est écrit, la version utilisée par Deno est la version 7.9.304 datant du 14 octobre 2019.

# ÉCOSYSTÈME & PREMIERS DÉVELOPPEMENTS

## Installation

Depuis quelques versions maintenant, Deno est disponible via Scoop pour Windows, et via Homebrew pour macOS. L'installation peut aussi être faite manuellement via cURL sous Shell, notamment pour Linux qui n'a que cette solution pour l'instant, ou via iwr sous PowerShell pour Windows.

Dans la même philosophie que le code, Deno est livré sous la forme d'un exécutable unique.

```
# Shell
curl -fsSL https://deno.land/x/install/install.sh | sh

# PowerShell
iwr https://deno.land/x/install/install.ps1 -useb | iex

# Scoop
scoop install deno

# Homebrew
brew install deno
```

Une fois l'installation terminée, lancer la commande deno https://deno.land/welcome.ts pour tester son bon fonctionnement.

## deno-cli

L'interface de ligne de commande propose un ensemble de fonctionnalités intégrées permettant de rester de manière immersive dans l'environnement de développement propre à Deno. Il permet aussi et surtout de rester aligné sur les standards lorsque l'on a besoin de proposer sa librairie à la communauté.

Voici une liste des commandes actuellement disponibles :

- deno info permettant d'inspecter les dépendances d'un programme à partir de son point d'entrée
- deno fmt permettant de formater le code grâce un prettier intégré
- deno bundle évoqué plus tôt, permettant de transpiler son application en un livrable unique avec les dépendances, en un fichier .js (donc utilisable par le navigateur)
- deno install permettant d'installer une application Deno dans son

~/.deno/bin depuis une URL ou un code local.

- deno types permettant de générer les types TypeScript de Deno pour le développement
- deno test permettant d'exécuter l'outil de test intégré. (Deno intègre en effet sa propre librairie de test)
- deno completions permettant d'ajouter la complétion automatique dans son terminal (normalement déjà ajouté lors de l'installation de Deno)
- deno eval permettant d'interpréter un fichier ou une chaîne de caractères contenant du code exécutable par Deno
- deno xeval (nommé sur la même pensée que xargs) permettant comme deno eval d'interpréter du code, mais en prenant en compte chaque ligne venant de stdin

## "HelloWorld.ts"

Parlons maintenant de notre premier programme. À l'heure actuelle, même si l'écosystème Deno lui-même propose un panel d'outils de développement utilisables en ligne de commande, le catalogue d'extensions VSCode (ou autre éditeur) reste très pauvre en fonctionnalités. Ne vous attendez pas à une expérience développeur complète pendant vos premières lignes de code.

### Exemple 1 : Grep

Ce premier exemple est une reproduction simple du comportement de *grep* et met en avant l'import des librairies standards Deno, leurs utilisations, ainsi que la manipulation de fichiers et d'arguments. Afin de les rassembler, les dépendances peuvent être déclarées dans un fichier conventionnellement appelé deps.ts:

```
import * as path from "https://deno.land/std/fs/path/mod.ts";
export { path };
export { green, red, bold } from "https://deno.land/std/colors/mod.ts";
```

Puis être importés classiquement dans son mod.ts (équivalent à l'index.js en Node):

```
import { path, green, red, bold } from "./deps.ts";
```

Un import "*http*" de Deno est une récupération de ressource Web au moment de la compilation. Deno supporte actuellement uniquement les protocoles http://, https://, et file://.
Ensuite, nous validons les arguments passés et récupérés directement depuis l'objet global Deno:

```
if (Deno.args.length != 3) {
  if (Deno.args.length > 3) {
    throw new Error("grep: to much args.");
  } else {
    throw new Error("grep: missing args.");
  }
}

const [, text, filePath] = Deno.args;
```

Enfin, nous parsons et itérons le fichier afin de faire ressortir les lignes contenant le *pattern* recherché:

```
try {
  const content = await Deno.readFile(path.resolve(Deno.cwd(), filePath));

  let lineNumber = 1;
  for (const line of new TextDecoder().decode(content).split("\n")) {
    if (line.includes(text)) {
      console.log(
        `${green(`(${lineNumber})`)} ${line.replace(text, red(bold(text)))}`
      );
    }
    lineNumber++;
  }
} catch (error) {
  console.error(`grep: error during process.\n${error}`);
}
```

Enfin, pour lancer l'application, exécutez la commande deno grep/mod.ts foo grep/test.txt
foo étant le *pattern*, et test.txt un fichier contenant des chaînes de caractère.

### Exemple 2: Overkill Gues-A-Number

Ce deuxième exemple est un mini jeu : le but est de trouver un nombre entre 0 et 10 à partir d'indices "plus" ou "moins". Il met en avant l'utilisation d'un framework tiers, de l'import de React, et de la compatibilité JSX.
L'import d'un tiers est presque identique à celui d'un standard :

```
import Home from "./page.tsx";
import {
  Application,
  Router,
  RouterContext
} from "https://deno.land/x/oak/mod.ts";
import { App, GuessSafeEnum, generate, log } from "./misc.ts";
```

Un fichier .tsx étant importé, React doit être utilisé pour pouvoir faire fonctionner l'ensemble. Le fichier page.tsx est donc complété ainsi :

```
import React from "https://dev.jspm.io/react";
import ReactDOMServer from "https://dev.jspm.io/react-dom/server";
```

On peut donc grâce à l'extension .tsx ainsi qu'à React utiliser JSX pour exporter un composant rendu côté serveur par exemple :

```
export default (props: HomeProps = {}) => `<!DOCTYPE html>
  ${ReactDOMServer.renderToString((
  <>
    <Home {...props} />
    <hr />
    <Debug {...props} />
  </>
))}`;
```

Vous pouvez lancer cet exemple via la commande deno guessanumber/mod.ts
Enfin, vous pouvez retrouvez les exemples complets sur Github ou même les exécuter directement depuis leurs URL "*raw.githubusercontent*" :
https://github.com/bios21/deno-intro-programmez

## PRODUCTION & FUTUR

Pour l'instant, Deno n'est pas *ready-to-prod*. Les principales utilisations étant de créer des outils de ligne de commande, des

gestionnaires de tâche de fond, ou des serveurs Web (à l'instar de *Node*), les performances de Deno ne sont pas au rendez-vous souhaité par Dahl. Il est toutefois possible de commencer à expérimenter grâce au développement d'outils internes comme des *batch scripts* par exemple. Un benchmark en temps réel est disponible sur https://deno.land/benchmarks.html

Commit après commit, les benchmarks sont mis à jour et comparent les performances de Deno à celles de Node sur plusieurs niveaux, par exemple le nombre de requêtes par seconde (qui est le premier goulot d'étranglement bloquant l'utilisation en production), la latence maximale, les interactions entrée-sortie, les consommations mémoire, etc. Deno est déjà meilleur que Node sur quelques points et ne cesse de s'améliorer au fil du temps pour espérer finir premier sur l'ensemble des tests effectués.

## Et la v1.0 ?

En plus des performances, Deno complète l'expérience développeur grâce à un ensemble de fonctionnalités et d'outils essentiels à la sortie de la version 1.0 qui pourra être considérée prête pour une utilisation en production.

### Debug

Il n'est pour l'instant pas possible de déboguer ou d'inspecter une application ; quelque chose qui peut être contraignant lors du développement. Cette fonctionnalité majeure est obligatoire pour la version 1.0. Profitant de V8, le debug se reposera sur le V8InspectorClient et les *Chrome Devtools* permettant d'utiliser les mêmes outils qu'avec n'importe quel autre développement JavaScript.

## COMMUNAUTÉ & CONTRIBUTION

Du fait de sa jeunesse, la communauté Deno reste encore petite. Néanmoins elle s'agrandit de jours en jours et nombreux sont les développeurs venant de Rust ou Node qui s'intéressent à la technologie.

Les communautés les plus grosses aujourd'hui sont Polonaises (qui comporte un des contributeurs majeurs en la personne de **Bartek Iwa czuk (@biwanczuk)** ), Coréennes, Chinoises ou encore Japonaises.

Des groupes de meetup naissent au fur et à mesure comme **Deno Poland (@denopoland)**, ou **Denoland Korea (@denoland_kr)**.

La France n'en est pas en reste et à déjà son premier groupe, **Paris Deno (@ParisDeno)**.

Une *newsletter* est aussi disponible sur https://deno.news

D'un point de vue de la contribution, il y a beaucoup à faire. Les *pull requests* sur les dépôts officiels sont "simples" à faire, car une liste de fonctionnalités manquantes et de bugs est disponible sur https://github.com/denoland/deno/milestone. De plus, les règles de contributions ont été écrites et complétées pour l'occasion.

La couche TypeScript comporte un core, un ensemble de librairies standards deno_std (https://deno.land/std/README.md), et un ensemble de librairies tierces regroupées en un répertoire unique pour en simplifier les URL (https://deno.land/x/).

Les contributions effectuées sur le standard et le *core* **doivent** respecter les règles. Ce n'est pas le cas pour les librairies tierces.

Les contributions peuvent aussi être faites au niveau des outils de développement. En effet, il manque encore beaucoup de chose pour être à l'aise et productif comme des extensions VSCode ou des librairies de test équivalentes à Jest ou fast-check (qu'elles soient portées, "isomorphisées", ou réécrites).

Deno a besoin de vous, n'hésitez pas à vous lancer et à soumettre votre contenu ; beaucoup de librairies proposées sont des portages de librairies existantes venant des API ou librairies Node, Rust, ou encore Go.

### Stabilisation de l'API

Il existe et subsiste encore quelques bugs dans l'API, que ce soit dans la couche TypeScript ou dans le deno-core. Ces bugs, quoique mineurs, restent néanmoins bloquants quant à la bonne stabilité de l'ensemble. Être stable ne signifie pas seulement avoir une exécution lisse, mais aussi avoir des points d'entrée cohérents et uniformes. Certaines fonctions doivent donc être revues au niveau de leur nom ou même de leurs signatures.

### Une documentation claire et explicite

Le problème commun à n'importe quel projet commençant dans l'ombre. La documentation de Deno est encore très légère et manque de cas d'usage ou d'explications sur des sujets précis.

Le site officiel est actuellement en cours de remodelage et ne tardera pas d'être complété.

## Futur

Décorrélés de la première version stable, des ajouts à la *CLI* seront faits, des supports sur l'ajout de fonctionnalités natives (via des modules appelés *"ops" crates* en Rust) seront apportés, ainsi, entre bien d'autres choses, que des compatibilités de plus en plus étroites avec le monde Web et les standards *ECMA* (en supportant par exemple les modules *WebAssembly*). Concernant la *CLI*, voici une liste non exhaustive des fonctionnalités prévues :

- deno compile permettant de compiler l'ensemble de son application en un binaire purement indépendant.
- deno doc permettant de générer une structure JSON de l'ensemble de la documentation du code. Ce JSON sera donc standard à Deno et pourra ensuite être consommé par un outil de documentation visuel comprenant ledit standard.
- deno ast permettant de générer une structure JSON de l'*Abstract Syntax Tree (AST)* du code à partir d'un point d'entrée donné. L'*AST* peut être consommé par des outils comme ESLint pour analyser la structure du code de manière programmatique et relever, par exemple, des potentielles failles dans le code ou des fuites mémoires.
- deno lint qui, en combinaison de deno fmt, permettra de rendre consistant entre les tous les développeurs le code produit et aussi d'en améliorer la qualité en l'alignant sur les standards de Deno. À retenir que la configuration du *linter* pour l'instant ne sera ni accessible ni modifiable.

La version 1.0 est très proche et le rythme soutenu des développements a permis à l'équipe d'estimer une sortie pour fin janvier, peu avant ou peu après la sortie de cet article.

Il est important de rappeler que Deno reste un projet *open source* et communautaire, et qu'il ne tient qu'à la communauté d'aider en expérimentant la technologie, en la poussant à ses limites, et en remontant le maximum de données aux développeurs.

Pour conclure, Deno n'en est qu'à ses balbutiements, mais Ryan Dahl n'est pas à son coup d'essai. Grâce à l'arrivée des nouvelles fonctionnalités de la version 1.0, la facilité d'utilisation de TypeScript, les performances de plus en plus intéressantes, et enfin et surtout, grâce à la communauté confiante et grandissante, Deno deviendra sans aucun doute une des potentielles *trending technologies* sur laquelle capitaliser pour 2020/2021.

Restez à l'écoute !

**Vivien Fabing**
*Lead Devops, Infinite Square*
http://blogs.infinitesquare.com

INFINITE SQUARE

# Blazor Webassembly : l'autre Blazor

*Imaginez pouvoir réutiliser vos connaissances en C# pour écrire une SPA (Single Page Application) ? Et bien ce rêve est maintenant réalité grâce à l'arrivée de Blazor WebAssembly, par la team aspnetcore ! Profitant de l'arrivée récente du nouveau standard WebAssembly, Blazor (l'abréviation de Browser + Razor) s'appuie actuellement sur le portage de Mono pour créer une application web. Voyons ensemble ce qui se cache derrière cette nouvelle techno.*

## Qu'est-ce que Blazor ?

Blazor, Blazor Server, Blazor WebAssembly, etc... Nous n'avons même pas encore écrit une seule ligne de code que l'on se retrouve déjà perdu entre différentes versions de Blazor !

En général, lorsque l'on parle de Blazor, on parle de la version Blazor WebAssembly, qui est celle qui exécute du code C# directement depuis le navigateur web, comme une SPA classique comme React, Angular, Vue.js, etc.

Dans cette version, vos dlls .NET seront envoyées directement à votre navigateur web et exécutées depuis celui-ci, comme n'importe quel fichier statique.

Blazor WebAssembly *est pour l'instant disponible en* Preview *dans la release 3.0 de .NET Core.*

De l'autre côté, vous avez peut-être entendu également parler de Blazor Server, qui est basé sur la technology SignalR et propose de conserver toute l'exécution de votre app côté serveur (*sans exposer ainsi vos dlls côté client*).

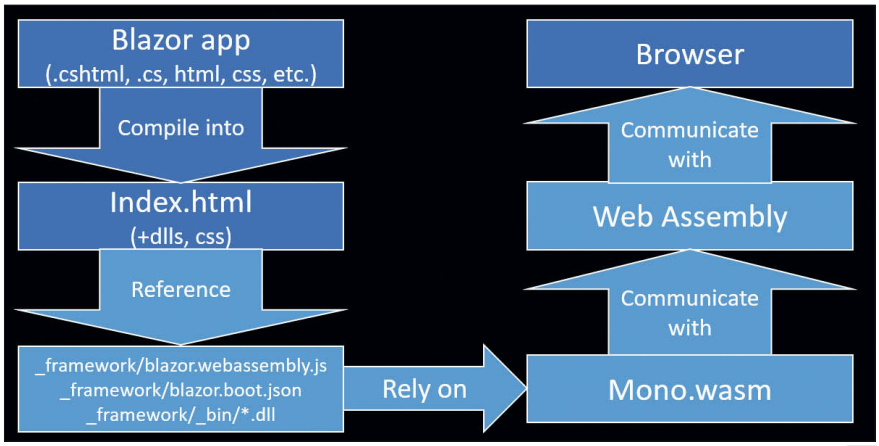
Blazor Server *est actuellement disponible dans la release 3.0 de .NET Core et est d'ores et déjà prêt à être utilisé en production.*

### Blazor WebAssembly

Tout d'abord, voyons un peu plus en détail Blazor WebAssembly.

Comme son nom l'indique, Blazor repose sur la syntaxe Razor pour générer votre application web. C'est pour cela que nous aurons besoin d'écrire des fichiers .razor et .cs, ainsi que les classiques fichiers .css pour la partie design. Un exemple de composant Blazor :

```
1  <div>
2    <h1>@Title</h1>
3
4    @ChildContent
5
6    <button @onclick="OnYes">Yes!</button>
7  </div>
8
9  @code {
10   [Parameter]
11   public string Title { get; set; }
12
13   [Parameter]
14   public RenderFragment ChildContent { get; set; }
15
16   private void OnYes()
17   {
18     Console.WriteLine("Write to the console in C#! 'Yes' button was selected.");
19   }
20 }
```




1

Vos fichiers seront ensuite compilés avec MSBuild pour générer des fichiers .css et .dll, reposant sur Mono et WebAssembly pour accéder au DOM et faire le rendu de votre application web.

Ci-dessus un rapide schéma décrivant ce comportement : 1

*Les sections en bleu clair sont des parties liées à l'infrastructure dont vous ne devriez idéalement pas avoir à vous soucier :)*

*À noter que l'architecture décrite ci-dessus s'avère être particulièrement adaptée dans des scénarios de développement où il est nécessaire de régénérer et déboguer facilement notre application, et devrait être en grande partie conservé dans les versions futures de* Blazor.

*En ce qui concerne les scénarios de production, le fonctionnement devrait changer et permettre une compilation AoT (Ahead of Time) qui devrait permettre à son tour de générer directement nos binaires en fichiers* .wasm *et d'obtenir ainsi des performances bien plus rapides.*

Au final, ce que vous obtenez en sortie de la compilation est une poignée de fichiers statiques, que vous pouvez tout à fait héberger comme n'importe quel site web statique classique en pur HTML, CSS Javascript !

À partir de là, tout est possible : de l'hébergement sur de l'Azure Blob Storage, au simple conteneur Docker Nginx, en passant par un hébergement standard sur de l'aspnetcore, tous les choix s'offrent à vous ! Toujours pas convaincu ? Ayant obtenu une applica-

tion web statique, vous accédez également à tout les outillages modernes du développement web :

- Vous voulez exécuter votre app Blazor comme une Progressive Web app pour avoir le support de l'offline, etc. ?
- Que diriez-vous de l'exécuter en mode **desktop** embarqué dans de l'Electron ?

Sans oublier de mentionner l'exécution de votre app en mode hybride sur du **mobile** avec Cordova

Ok, tout ça semble très expérimental me direz-vous. Mais vous savez quoi ? Le meilleur dans tout ça, c'est que l'équipe aspnetcore est également en train de suivre de près ces nouveaux usages et prévoit de les inclure dans leur Roadmap pour que tout ça vienne en standard avec aspnetcore ! Ah oui, j'oubliais le point le plus évident : étant donné que notre application Blazor est écrite en **C#**, le partage de code entre les parties clientes et serveurs sont évidemment très simples, et l'on obtient accès aux packages **NuGet** également, etc.! Si je pense que l'utilisation principale de Blazor se fera principalement via sa version Blazor WebAssembly, voyons tout de même un scénario plutôt inattendu mais néanmoins très intéressant nommé Blazor Server.

## Blazor Server

Envie de conserver un client léger et une charge plutôt côté serveur, tout en fournissant une interface graphique qui peut se mettre à jour dynamiquement sans rechargement de page ?

Blazor Server, basé sur SignalR, propose que le client se contente de télécharger un petit JavaScript pour se connecter au système SignalR, et ensuite envoie tous les événements JavaScript au serveur sur lequel le DOM virtuel est calculé, qui se charger enfin de renvoyer au client les modifications du DOM à effectuer.

Il y a quelques avantages pour cette méthode :

- Les fichiers .dll ne sont pas accessibles depuis le navigateur web.
- L'exécution de dotnetcore reste côté serveur, et permet d'accéder plus facilement aux fonctionnalités côtés serveur (*plutôt que les fonctionnalités du navigateur web*)
- *Le support d*'IE 11...

Cependant, vous n'êtes alors plus dans une SPA et vous perdez les bénéfices suivants :

- Plus de déploiement "à la fichiers statiques" car un serveur aspnetcore est requis,
- Plus de PWA car vous avez besoin d'être constamment connecté au serveur
- Chaque événement JavaScript est calculé côté serveur plutôt que côté client, générant du coup un peu de latence.

Autre avantage à l'heure actuelle : Blazor Server est la seule version supportée en production ! Du coup démarrer un projet en Blazor Server permet de prendre de l'avance sur la préparation de ses composants Blazor, afin de pouvoir basculer en Blazor WebAssembly lors de sa sortie en version stable.

# Commencer à développer une application Blazor en quelques secondes

Aujourd'hui, si vous voulez démarrer un projet Blazor en quelques secondes, vous pouvez simplement suivre la documentation officielle, qui est composée de 3 étapes principales :

- Installer la dernière version 3.0 du SDK .NET Core
- Installer la version la plus récente du template Blazor en exécutant la commande dotnet new -i Microsoft.AspNetCore.Blazor.Templates::3.0.0-*
- Créer un nouveau projet Blazor WebAssembly en exécutant les commandes :
  - dotnet new blazorwasm -o MyBlazorWebAssemblyProject
  - puis cd MyBlazorWebAssemblyProject
  - et enfin dotnet run

Et c'est tout, vous obtenez alors une application Blazor WebAssembly qui fonctionne, et à partir de laquelle vous pouvez commencer à développer.

Pour la déployer en production, il suffit d'exécuter la commande dotnet publish -c release puis de copier le contenu du dossier publish/dist sur votre hébergeur de site statique préféré, et voilà !

Mais si comme moi, vous n'aimez pas trop partir du principe que "tout marche, comme par magie", continuez de lire la suite de cet article :)

## Le projet Blazor le plus simple possible

Pour créer une application Blazor la plus simple possible, nous allons tout d'abord avoir besoin d'un fichier de projet C# .csproj. Celui-ci va permettre de définir les versions du tooling spécifique à Blazor, ainsi que les dépendances nécessaires (Microsoft.AspNetCore.Blazor, Microsoft.AspNetCore.Blazor.Build et Microsoft.AspNetCore.Blazor.HttpClient)

Ensuite, comme n'importe quel projet ASP.NET core classique, vous aurez besoin d'un fichier Program.cs qui appellera un fichier Startup.cs. Ce dernier sera en charge de référencer le composant racine Blazor, appelé par convention App.

Puis, pour la partie purement Blazor, vous aurez besoin d'un fichier App.razor définissant le composant racine, ainsi qu'une page par défaut, généralement appelée Index.razor, que l'on placera à l'intérieur d'un dossier Pages (tout ça par convention).

Pour finir, ce qui permettra de lier tout ça est un fichier index.html, contenu dans le dossier wwwroot. Dans ce fichier, on y trouvera notamment une référence vers le framework blazor.webassembly.js qui permettra de charger le runtime mono dans sa version webassembly, ainsi que toutes vos librairies .NET (.dll), elles-mêmes définies dans un fichier blazor.boot.json.

Ci-dessous un petit récapitulatif de tous les fichiers mentionnés :

- Un fichier .csproj de projet C#,
- Un fichier d'entrée Program.cs,
- Un fichier Startup.cs,
- Un composant Blazor racine défini dans un fichier App.razor,
- Une page Index.razor définie dans un dossier Pages (*par convention*),
- Puis finalement un fichier index.html à l'intérieur du dossier wwwroot qui sera notre point d'entrée.

Vous pouvez voir le contenu de ces fichiers dans le commit Github #ADD smallest publishable static Blazor project, mais à ce stade, ils sont principalement vides.

Pour déployer votre projet, il suffit d'exécuter la commande dotnet publish -c release puis de récupérer le contenu du dossier ./bin/release/netstandard2.0/publish/yBlazorWebAssemblyProject/dist pour le déposer dans votre hébergeur de site statique préféré. 2

L'application Blazor en tant que telle nécessite de télécharger 5,1 mo. Cela est principalement dû à la taille des fichiers mono.wasm

et mscorlib.dll qui pèsent respectivement 1,8 mo et 1,3 mo.
Mais puisque l'application consiste uniquement en de simples fichiers statiques, ils ne devraient idéalement être téléchargés qu'une seule fois, permettant d'avoir uniquement quelques ko à télécharger lors des futurs chargements de l'application :)

## Où héberger une application Blazor ?

Concernant l'hébergement, en fait la documentation officielle décrit plutôt bien les différentes possibilités qui s'offrent à nous, mais j'aimerais revenir rapidement sur ces options et bien sûr y apporter mon grain de sel :)

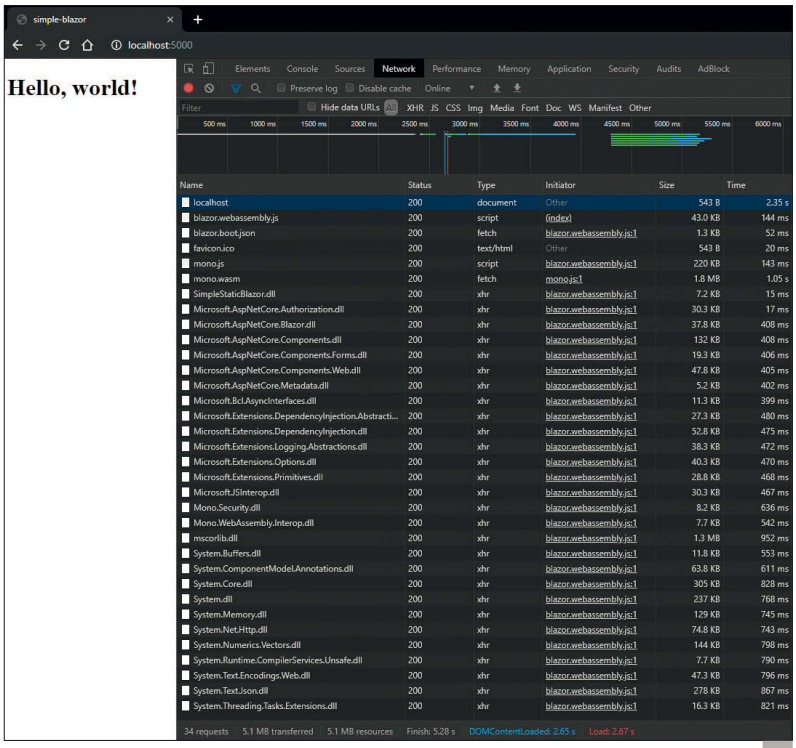
### Hébergement Blazor sur un serveur web de site statique

Si vous êtes déjà familier avec le bon vieux serveur IIS, Blazor offre une intégration "out-of-the-box", et fournit le fichier web.config nécessaire pour l'hébergement, retrouvable directement parmi les fichiers de publication de notre application.
Par contre comme notre application Blazor est une SPA, nous aurons besoin d'installer le URL Rewrite module afin de rediriger nos requêtes http vers notre application (*e.g.: rediriger la requête* /about *vers le fichier index.html contenant notre SPA plutôt que d'essayer de trouver des fichiers dans un potentiel dossier IIS* about)
Et si vous avez plus l'habitude de manipuler des serveurs Nginx ou Apache, vous pouvez bien évidemment héberger votre application Blazor dessus. Vous pourrez trouver des exemples de configuration simple de fichier nginx.conf ou de configuration Apache sur la documentation officielle.
Mais poursuivons un peu plus en détail cet hébergement Nginx, notamment à l'intérieur d'un conteneur Docker.

### Héberger votre SPA dans un simple conteneur Nginx

En fait, vous avez déjà un exemple de Dockerfile sur la documentation officielle (*du bon boulot cette doc, vraiment* :), mais, probablement pour des raisons de simplicité, le fichier Dockerfile assume que vous êtes déjà en possession d'un package prêt à être déployé, pour lequel il ne resterait plus qu'à être copié dans le conteneur.
Si cette approche va fonctionner techniquement, je préfère beaucoup plus l'idée d'avoir un fichier Dockerfile à côté de ma solution, capable de le builder et de générer les fichiers nécessaires pour exécuter mon application.
Pour arriver à ce scénario, nous allons reproduire le même mécanisme qui est fourni par Visual Studio pour conteneuriser une application aspnetcore : utiliser une Build Docker multi-stage qui se basera sur l'image du sdk dotnetcore et sur une petite image Nginx.
*Note : étant donné que Blazor n'est apparu que depuis la version 3 du framework dotnet core, il est évidemment nécessaire de récupérer une image possédant au minimum cette version.*
Voilà à quoi ressemble notre Dockerfile :

```
1  FROM nginx:alpine AS base
2
3  FROM mcr.microsoft.com/dotnet/core/sdk:3.0 AS publish
4  WORKDIR /src
5
6  COPY ["SimpleStaticBlazor.csproj", ""]
7  RUN dotnet restore
8
9  COPY . .
10 RUN dotnet publish -c release -o /app
11
12 FROM base AS final
13 COPY nginx.conf /etc/nginx/nginx.conf
14 COPY --from=publish /app/SimpleStaticBlazor/dist /usr/share/nginx/html/
```




3

Plutôt simple et concis non ?
On a ensuite besoin d'effectuer les commandes docker build -t simplestaticblazor . (*ne pas oublier le "." à la fin*) pour builder notre image, ainsi que la commande docker run -it --rm -p 5000:80 simplestaticblazor pour le démarrer et pouvoir y accéder à l'url http://localhost:5000

### Exposer votre SPA Blazor depuis Azure Blob Storage

Si vous êtes déjà habitué à travailler avec Azure, ce mode de fonctionnement devrait être des plus simples : sur le service Azure Blob storage, il existe une fonctionnalité de Static website qui permet d'exposer les fichiers contenus dans le stockage en http : 3
Une fois activée, il suffit juste de copier nos fichiers statiques dans le compte de stockage via notre outil préféré tel qu'Azure Storage Explorer ou encore la tâche Azure Devops Azure File Copy, etc.
Et voilà, vous obtenez une application plutôt scalable sans avoir besoin de gérer un service plus complet/complexe, tel qu'Azure app service, juste pour héberger une poignée de fichiers statiques.
Et pour aller jusqu'au bout, il ne faudra pas oublier d'utiliser un Azure CDN afin d'y associer un nom de domaine custom exposé en https.

## Héberger notre SPA sur Github pages

Pour héberger notre SPA Blazor sur Github pages, nous avons besoin d'une petite astuce que j'ai trouvée tellement sympa que je ne pouvais pas ne pas en parler :)

Le problème sur Github pages étant qu'il n'est pas possible de rediriger toutes les requêtes vers un seul et même fichier, car le service ne propose que 2 manières de gérer la redirection lorsque le visiteur tente d'accéder à une url ne correspondant pas à un fichier :

- Soit l'utilisateur tente d'accéder à la l'Url racine du site web, et c'est la page par défaut index.html qui est retournée.
- Soit un fichier 404.html est renvoyé à l'utilisateur plutôt que de lui retourner une erreur 404 not found.

Heureusement, cette limitation est bien connue par la communauté qui a rendu disponible 2 petits hacks JavaScript : Ceux-ci permettent de rediriger depuis la page 404 vers la page index.html avec les information de routage, et permettent d'avoir une application Blazor fonctionnelle sur Github pages.

Je ne m'attarde pas plus sur cette méthode, mais je pense qu'elle valait le coup d'être mentionnée !

## ASP.NET Core loves Blazor

Malgré tout, notre meilleure option pour héberger notre app Blazor reste depuis une application aspnetcore (et donc par-dessus un serveur Kestrel)

En effet, aspnetcore possède déjà tout un tas de petites extensions qui permettent d'héberger notre application Blazor, de la déboguer, lui rediriger les requêtes, etc.

Exemple du Startup.cs de notre application aspnetcore hébergeant une application Blazor :

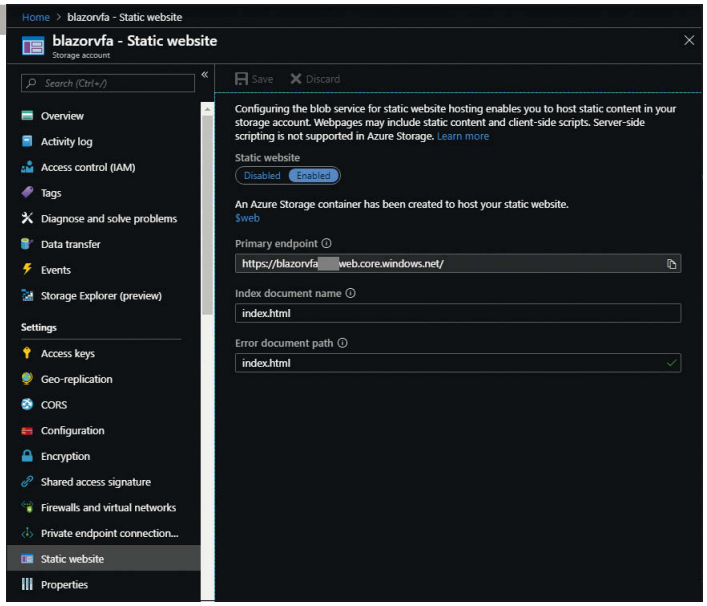
```
1  public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
2  {
3      app.UseResponseCompression();
4
5      if (env.IsDevelopment())
6      {
7          app.UseDeveloperExceptionPage();
8          app.UseBlazorDebugging();
9      }
10
11     app.UseStaticFiles();
12     app.UseClientSideBlazorFiles<Client.Startup>();
13
14     app.UseRouting();
15
16     app.UseEndpoints(endpoints =>
17     {
18         endpoints.MapDefaultControllerRoute();
19         endpoints.MapFallbackToClientSideBlazor<Client.Startup>("index.html");
20     });
21 }
```

Ce type d'hébergement permet également des scénarios plus avancés tel que le Server Prerendering.

Le html de notre application Blazor est pré-calculé directement depuis le serveur, ce qui permet de renvoyer une simple page html lors de l'arrivée d'un premier visiteur, accompagnée d'un petit fichier JavaScript. Ce petit fichier Javascript permettra ensuite de "réhydrater" notre fichier html (i.e. reconnecter le html affiché et le dom virtuel) pour que notre visiteur obtienne au bout de quelques millisecondes une SPA Blazor complètement fonctionnelle !

Daniel Roth, Principal Program Manager sur Blazor, a déjà publié sur son Github une démo fonctionnelle de cette partie.

3




## Comment transformer une application Blazor en PWA avec Workbox

Si vous n'avez pas encore entendu parlé des PWA, ce sont des applications web qui s'approchent du look and feel des applications natives, permettant d'être installables, de fonctionner offline, d'envoyer des notifications push, etc.

Personnellement, j'adore l'idée de pouvoir fournir une application facilement installable sur n'importe quel système (Windows, Android, iOS) dans avoir besoin d'utiliser différents outils, langages, manières de déployer, etc… Grosso modo, nous allons avoir besoin de 2 fichiers pour créer notre PWA :

- un fichier manifest.json décrivant notre application installable (son nom, icônes, etc.)
- un fichier Service Worker sw.js pour gérer le cache offline, les notifications push, etc.

Voyons maintenant plus en détails comment les ajouter.

### Décrire l'installation de notre application Blazor avec un fichier manifest.json

Ajouter un fichier manifest.json à notre application Blazor est plutôt simple, on aura besoin de l'ajouter au dossier wwwroot afin qu'il soit publié dans le package final, ainsi que de rajouter une référence dans le fichier index.html.

Un fichier manifest.json ressemble à ça :

```
{
 "short_name": "blazor-workbox-pwa",
```

```
"name": "Blazor Workbox PWA",
"icons": [
 {
  "src": "/images/icons-192.png",
  "type": "image/png",
  "sizes": "192x192"
 },
 {
  "src": "/images/icons-512.png",
  "type": "image/png",
  "sizes": "512x512"
 }
],
"start_url": "/",
"background_color": "#3367D6",
"display": "standalone",
"scope": "/",
"theme_color": "#3367D6"
}
```

Il va nous permettre de définir le nom de notre app (short_name, name), un peu de cosmétique (icons, background_color, theme_color), l'URL à accéder en mode offline start_url, etc.
On aura également besoin de rajouter quelques assets supplémentaires tels que des icônes, favicon etc. pour avoir une jolie PWA, et enfin rajouter la ligne suivante dans la balise <head> de notre fichier index.html :

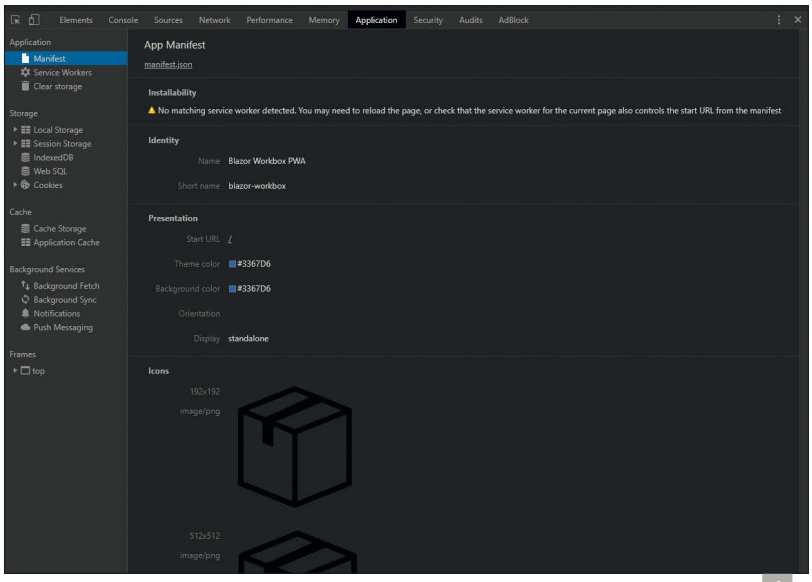
```
<link rel="manifest" href="/manifest.json">
```

Les Chrome DevTools devraient afficher quelque chose de similaire. 4

## Ajouter le mode offline à notre PWA Blazor avec Workbox

Un avantage que j'adore avec les applications Blazor est l'accès à tout l'outillage standard et mature des SPA classiques.
Et personnellement l'un des outils dont l'accès me rend particulièrement heureux s'appelle Workbox : Je ne sais pas si vous avez déjà essayé d'écrire un service worker avec du JavaScript Vanilla, mais personnellement ce n'était clairement pas ma meilleure expérience de développement :)
Du coup Workbox est un outil open source maintenu par Google qui simplifie grandement le développement de service worker.
En gros, il se compose d'un outil en ligne de commande et va nous permettre de générer notre fichier de service workersw.js, prêt à être référencé dans notre fichierindex.html`.
Tout d'abord, nous aurons besoin d'un fichier workbox-config.js similaire à celui-ci :

```
module.exports = {
  "globDirectory": "bin/Release/netstandard2.0/publish/blazor-workbox-pwa/dist",
  "globPatterns": [
    '**/*.{html,json,js,css,png,ico,json,wasm,dll}'
  ],
  "swDest": "bin/Release/netstandard2.0/publish/blazor-workbox-pwa/dist/sw.js"
};
```




4

Où :

- globDirectory permet d'indiquer à Workbox ou sont situés les fichiers à mettre dans le cache pour le mode offline. Dans notre cas, ça sera le dossier de sortie de la commande dotnet publish.
- globPatterns permet de filtrer les fichiers à prendre compte dans le cache tel que les fichiers html, css, js ainsi que les fichiers d'assets, mais surtout les fichiers wasm et dll car l'application en question est une application qui s'effectue avec le framework mono webassembly :)
- et pour finir, le paramètre swDest permet de préciser où placer notre fichier auto généré de Service Worker sw.js, dans notre cas le dossier de publication de notre application.

*Note : il existe d'autres clés de configuration disponibles avec ce fichier de configuration. Plus d'information disponible sur la* documentation officielle

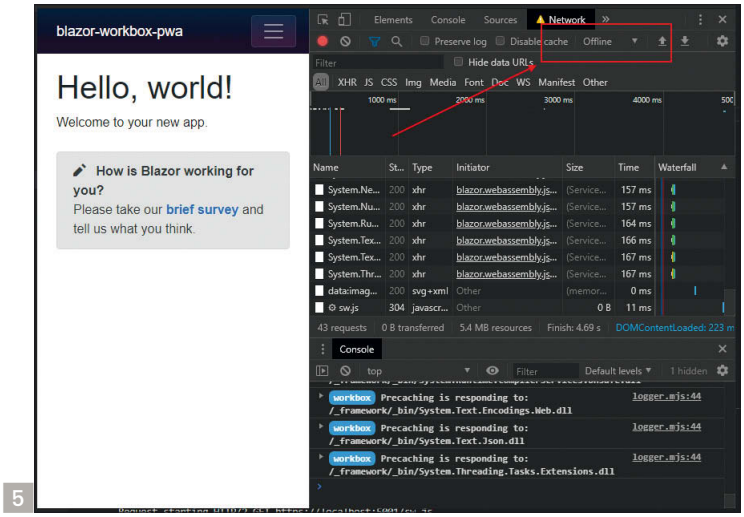
Enfin, il nous reste à ajouter une référence vers notre fichier de service worker dans notre fichier index.html en ajoutant les lignes suivantes à la fin de notre balise <body> :

```
<script>
  // Check that service workers are supported
  if ('serviceWorker' in navigator) {
    // Use the window load event to keep the page load performant
    window.addEventListener('load', () => {
      navigator.serviceWorker.register('/sw.js');
    });
  }
</script>
```

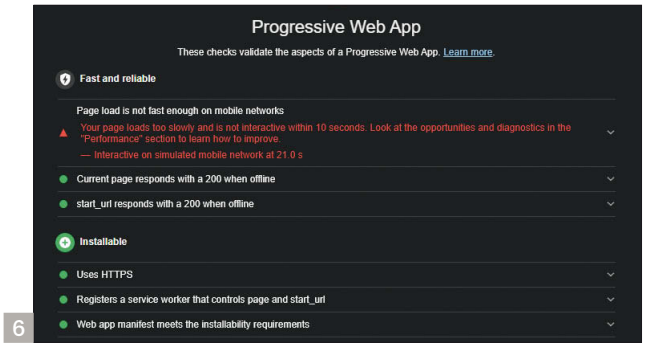
Pour générer notre package de production, on exécutera les commandes suivantes :

```
dotnet publish -c Release
workbox generateSW workbox-config.js
```

Ce package est bien entendu utilisable sur n'importe quel service d'application web statiques capable d'héberger une SPA…




5




6

Et voilà ! On obtient une PWA Blazor installable et utilisable même en offline ! 5

Remarquez l'option de réseau mise à offline et pourtant l'application reste fonctionnelle ! Dernière petite vérification, regardons ce que Chrome Lighthouse nous donne : 6

Pas mal pour une mise en route avec si peu de lignes de code :)

## Comment afficher une SPA Blazor webassembly plus rapidement grâce au prérendu serveur

Maintenant que nous avons obtenu une application qui fonctionne offline, voyons voir un peu c**omment afficher notre application Blazor webassembly plus rapidement au démarrage**.

En effet, lorsque l'application est composée uniquement de fichiers statiques, tous les fichiers ont besoin d'être chargés avant de pouvoir afficher le rendu graphique, ce qui affiche un message "Loading..." à nos utilisateurs lors du premier accès à l'application.

Par contre, une fois les fichiers chargés, notre SPA reste bien entendu super rapide car elle n'a plus besoin de régénérer l'intégralité des pages sur notre serveur.

Maintenant, si vous vous souvenez de ce "bon vieux temps" où l'on faisait de l'aspnet MVC (comment ça ce temps n'est pas si vieux que ça ? :)), recharger l'intégralité de la page lors de chaque action est une surcharge qui ne nous manque pas le moins du monde. Mais au moins, le premier affichage de notre application web n'avait pas besoin de temps de chargement !

Si seulement nous pouvions avoir un système intelligent, avec un premier rendu effectué une première fois depuis le serveur, mais qui récupérerait ensuite tous les fichiers pour fonctionner en mode SPA en tâche de fond... Et bien ce temps-là est arrivé ! Réjouissons-nous d'une fonctionnalité super basique mais pourtant si complexe pour notre SPA : Le Server Prerendering ! (*Prérendu serveur*)

En fait, le fonctionnement de Blazor en mode Server side Blazor (qui est d'ailleurs le seul mode de fonctionnement supporté en production à l'heure actuelle), utilise déjà ce rendu de composants Blazor côté serveur, puis utilise ensuite SignalR pour envoyer les modifications du DOM au navigateur web.

Mais voyons plus en détail comment ajouter ce prérendu serveur à notre SPA Blazor !

## Est-ce que le prérendu serveur n'a que des avantages ?

Et bien pas vraiment. Sans même avoir besoin de parler des problématiques et limitations techniques engendrées par l'ajout du prérendu serveur (*du genre "double déclenchement de l'événement* OnInitialized, *etc.*), l'un des points qui personnellement me marque le plus, c'est d'avoir à héberger notre application dans une application aspnetcore.

Même si en tant que développeur .NET, ça n'est pas vraiment un problème (*voire ça pourrait même être un avantage*), cela a quand même pour effet de réduire les possibilités d'hébergement de notre application (*Le stockage dans de l'Azure Blob Storage n'est plus possible par exemple*), et l'on aura besoin d'embarquer le runtime aspnetcore avec notre application pour qu'elle fonctionne (*point qui est heureusement facilité avec les technologies de conteneurs telles que Docker*)

## Prérendu serveur d'une application Blazor webassembly avec aspnetcore

C'est parti, il est temps de rentrer un peu plus dans la technique ! Tout d'abord, nous aurons besoin d'une application aspnetcore 3, à créer avec votre IDE favori, ou via la ligne de commande dotnet new web.

On aura besoin de 2 choses :

- Ajouter une référence vers le package Nuget Microsoft.AspNetCore.Blazor.Server,
- et ajouter une référence vers notre projet Blazor.

Ensuite, on ajoute un fichier _Host.cshtml dans un dossier Pages afin de faire le rendu de notre composant Blazor racine App directement depuis le serveur, un peu "à la manière" de Razor pour faire le rendu de pages web. Le contenu de ce fichier devrait être identique à notre fichier index.html, modulo le remplacement de la balise affichant le message "Loading..." par la ligne suivante :

```
1  @using blazor_workbox_pwa
2  ...
3  <app>@(await Html.RenderComponentAsync<App>(RenderMode.ServerPrerendered))</app>
```

Ensuite dans le fichier Startup.cs de notre application aspnetcore, on a besoin d'ajouter dans la méthode ConfigureServices les services nécessaires au rendu de notre page "Mvc", ainsi le HttpClient utilisé par notre application Blazor :

```
1  services.AddMvc();
2  services.AddScoped<HttpClient>(s =>
3  {
4    var navigationManager = s.GetRequiredService<NavigationManager>();
```

```
5   return new HttpClient
6   {
7     BaseAddress = new Uri(navigationManager.BaseUri)
8   };
9 });
```

Enfin dans la méthode Configure, on a besoin d'ajouter les configurations suivantes :

```
1  if (env.IsDevelopment())
2  {
3    ...
4    app.UseBlazorDebugging();
5  }
6  else
7  {
8    app.UseHsts();
9  }
10 app.UseHttpsRedirection();
11 app.UseClientSideBlazorFiles<blazor_workbox_pwa.Startup>();
12 app.UseStaticFiles();
13 app.UseRouting();
14 app.UseEndpoints(endpoints =>
15 {
16   endpoints.MapDefaultControllerRoute();
17   endpoints.MapFallbackToPage("/_Host");
18 });
```

On peut d'ores et déjà tester le prérendu de notre application en lançant l'application aspnetcore, et ainsi constater que le premier affichage de notre page n'affiche plus de temps de chargement : 7

Bon d'accord, ce n'est peut-être pas aussi "mind blowing" que ça, mais revoyons le fonctionnement initial sans prérendu serveur : 8

C'est mieux, non ?

## Sur le chemin de la PWA Blazor "idéale"

Au final, après quelques points d'optimisation (activation de la compression, ajout de balises ou attributs html nécessaires, etc. plus d'infos sur mon GitHub(1)), voyons voir notre score sur Lighthouse des Chrome DevTools : 9

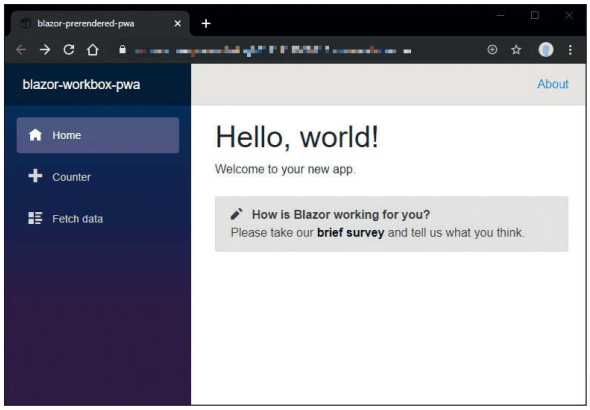
On obtient le score maximal concernant le côté PWA de notre application ! On a toujours quelques pistes d'amélioration côté performance (correspondant au score de 85), principalement du à la taille totale de toutes nos dlls, nécessaires pour interagir avec notre app.

À ce sujet, j'ai pas mal d'espoir que la future implémentation d'une fonctionnalité de compilation AoT (Ahead of Time) améliore les performances ainsi que la taille du package. Plus d'infos dans le ticket #5466 sur GitHub.

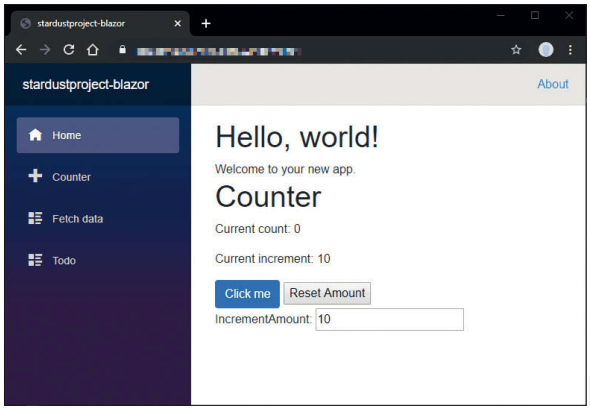
## En conclusion

Comme toujours, j'espère que cet article vous aura donné une bonne vue d'ensemble de Blazor Webassembly et des idées pour l'utiliser.

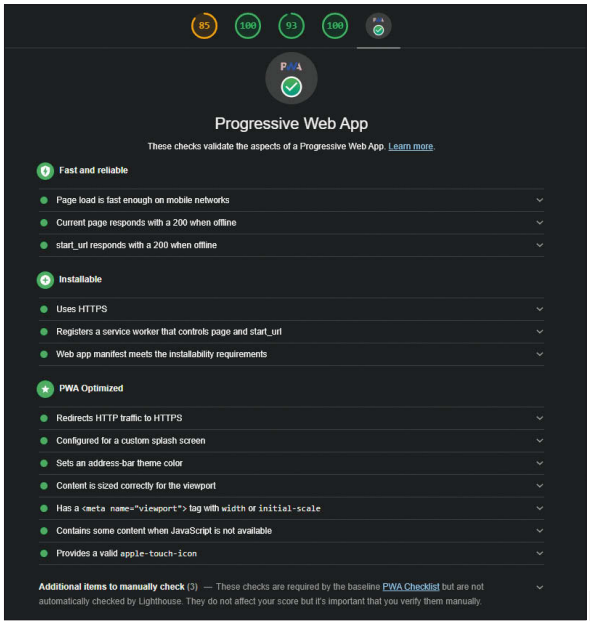
(1) https://github.com/vfabing/blazor-workbox-pwa/tree/9d237c936d0af3ee9a0a0a17d78547eeb96791d1




7




8




9

Merci beaucoup à Chris Sainty qui contribue beaucoup à la communauté Blazor, et notamment pour son article *Prerendering a Client-side Blazor Application* que je vous recommande de lire également ! Merci également à notre guru Blazor Daniel Roth qui avait partagé un exemple de prérendu serveur avec Blazor en premier-lieu sur GitHub !

N'hésitez pas à me joindre sur Twitter @vivienfabing, dans les commentaires ou par n'importe quel autre moyen, et que le code soit avec vous !

Frédéric POINDRON
Ukitechs
www.ukitechs.fr

# Blazor vs Angular : vrai débat ou idées reçues ?

*Frédéric POINDRON, architecte logiciel et gérant de Ukitechs, connaît Blazor et Angular. Quels sont les pour et les contre des deux technologies ? Même si la comparaison n'est pas toujours facile, quelques éléments peuvent éclairer les arguments. Mais finalement, rien ne remplacera les PoC pour juger par soi-même.*

## Les + de Blazor

**Productivité :**

- Un seul langage informatique à maitriser (C#) pour réaliser le front End (Client Web qui fonctionne dans un browser internet) et le back End (Serveur Web). Ainsi, un seul et même développeur peut participer indifféremment à la réalisation/correction de l'une ou l'autre des parties.
- Les équipes ne sont plus obligées d'avoir 2 types d'expert, l'un pour le Front end, et l'autre pour le Back end.
- Blazor intègre également un Framework de tests unitaires qui autorise le test automatisé des classes modèle, mais également (et c'est plus rare) de l'interface graphique Web !

**Réutilisation du code :**

- Puisque Back end et Front end sont codés dans le même langage, le partage de classe C# est autorisé. Cela évite la pénible tâche de coder 2 fois la même chose, et minimise drastiquement les risques liés ! Les classes d'échange entre front End et Back End (celles qui servent à véhiculer les données encodées JSon) sont partagées ; les modifications appliquées sont immédiatement disponibles dans le Front end comme dans le Back end. Pas besoin d'utiliser Nswag ou OpenApiGenerator pour générer le modèle front-end.
- Il en est de même pour **le code de validation métier** que l'on peut coder une fois, et réutiliser directement dans la partie « client Web » comme dans la partie Serveur. Il est assez facile d'utiliser les mêmes validations d'annotations de données côté serveur et client grâce à EditForm, un composant de Blazor :

```
using System.ComponentModel.DataAnnotations;

public class ExampleModel
{
  [Required]
  [StringLength(10, ErrorMessage = "Name is too long.")]
  public string Name { get; set; }
}
```

**Outils et habitudes :**

- Les outils pour coder Front end et Back end sont les mêmes : Visual Studio 2019
- Le débogage de la partie Front end se fait également via Visual Studio 2019 !
- Les tests unitaires sont réalisés de façon standard.
- Les patterns C# sont exploitables dans les 2 mondes.

De ce fait, la montée en compétence est rapide. Pour un développeur déjà habitué au mode .Net, c'est un réel gain de productivité !

## Ne pas sous-estimer la « haine » de JavaScript

JavaScript est aujourd'hui le **seul langage** autorisé pour réaliser la partie Front End (client Web). En dépit de ses nombreuses faiblesses, c'est un passage obligé. De ce fait, Javascript caracole aujourd'hui en tête des langages les plus utilisés… et il en est de même pour la section des langages les plus détestés. « *Nous utilisons JavaScript tous les jours dans nos applications. Nous sommes conscients des pièges et des choses sympas qu'il peut faire. Cela dit, la majorité de mon équipe de développeur souffre de ce langage, et ne demande qu'à en changer. L'arrivée de C# dans cette cours d'école constitue pour nous un réel bol d'air. Tout ce que je fais en JS, je peux aussi le faire en C#, et même en mieux. Il n'y a aucun débat.* »

## Les – de Blazor

- Côté serveur : Dans le cadre d'une utilisation SSB, il semble que Blazor puisse avoir des problèmes de montée en charge car il nécessite une connexion SignalR maintenue et constante (SignalR peut avoir des soucis de montées en charges en cas de forts trafics, NDLR).
- Côté client : L'utilisation de Blazor en mode CSB (WebAssembly) nécessite le support d'un navigateur dit moderne, donc récent. Ainsi, sa mise en œuvre via IE11 peut être problématique, en dépit des possibilités de « fall back » prévues. Angular, React, Node sont, par contre, être compatibles avec un IE11…

Réutilisation de classes métiers : des générateurs de code C#/JS existent.

## Conclusion

À l'identique de toute nouvelle technologie, l'adoption de Blazor sera progressive. D'abord évaluée sous forme de POC, elle sera réellement effective d'ici fin 2020.

Mais je pense qu'un usage « généralisé » en production prendra plusieurs années. Blazor est « juste » une technologie supplémentaire, un autre moyen de faire du client Web. Seule son utilisation sous forme de « WebAssembly » (CSB) constitue une réelle révolution. C'est là sa véritable plus-value.

Cependant, il ne faut pas sous-estimer le traumatisme causé suite à l'abandon de Silverlight : Blazor comme feu Silverlight sont des produits Microsoft, et nombreux sont ceux qui redoutent que Blazor ne finisse comme son prédécesseur. Après tout, aujourd'hui, ASP.Net couplé à React ou Angular fonctionne très bien, alors pourquoi en changer ?

Beaucoup d'obstacles et de problèmes techniques restent à franchir. Pourtant, je reste persuadé que Blazor saura les franchir, et s'imposera auprès des développeurs. Les développeurs maitrisant .Net/C# sont les premiers concernés. Le passage par JavaScript/TypeScript n'est désormais plus une obligation à la réalisation de clients Web, et de réels gains de productivité sont promis grâce à .Net et aux outils qui l'accompagnent !

**Alysson Hussin**
*Développeuse chez Adneom, bépoète, libriste. J'ai commencé à développer à l'âge de 14 ans en C et je n'ai jamais quitté le code depuis.*

# (Re)découvrir PHP à travers Laravel

***Quand on débute dans le monde du travail, on accepte souvent la première offre positive, quel que soit le langage. Peut-être que le langage de programmation demandé n'était pas votre coup de cœur. Peut-être aussi souhaiteriez-vous retourner vers votre premier amour, celui que vous avez lâché quelques années auparavant pour vous conformer au marché des technologies ? Si PHP était ce grand amour pour vous, peut-être seriez-vous intéressé(e) par Laravel.***

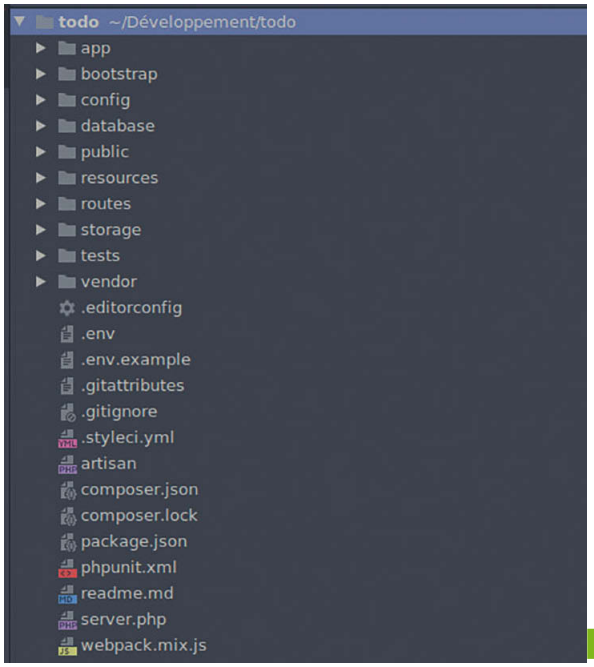
Laravel est un framework PHP où les bases sont déjà construites. La seule chose dont nous devons nous soucier est le côté business d'une application. Chaque élément et emplacement de vos fichiers .php sont destinés à une chose. Chaque élément clé d'une application web a son emplacement dédié : contrôleurs, vues, migrations, modèle, etc.

## Petite API pour comprendre

Commençons les choses sérieuses avec la création d'une API REST avec notre framework. Pour utiliser la dernière mouture de Laravel (6.x), il vous faudra PHP 7.2 ou + ainsi que d'autres dépendances : BCMath, Ctype, JSON, Mbstring, OpenSSL, PDO, Tokenizer et XML et le gestionnaire de paquets Composer qui orchestrera toutes les versions de dépendances de votre application.
Pour créer un projet avec Laravel, les commandes sont `composer global require laravel/installer` et ensuite `laravel new todo` pour installer Laravel en tant que tel sur votre ordinateur. Ou simplement :

```
composer create-project –prefer-dist laravel/laravel todo
```

Si vous souhaitez installer la dernière version du framework disponible même si elle n'est pas présente sur votre machine.
Quand vous ouvrirez le projet pour la première fois, vous tomberez sur cette architecture : 1



1

Expliquons les différents dossiers créés à la création du projet :

- **app**: dossier contenant vos contrôleurs, modèles, middlewares, commandes, exceptions et providers de votre application.
- **bootstrap**: c'est l'endroit où Laravel initialisera les outils de chargement de votre application et des dépendances tierces.
- **config**: il contient toute la configuration nécessaire au bon fonctionnement de l'application, directement lié au .env de vos fichiers, c'est par exemple ici que vous pourrez ajouter vos énumérations, etc.
- **database**: magnifique endroit où sont sauvegardés vos migrations et vos seeders. Nous verrons ensemble après à quoi peut bien servir le dossier factories.
- **public**: c'est là que votre application atterrira lorsque vous la mettrez en production.
- **Resources**: toutes les ressources de votre application que cela soit du JavaScript, votre configuration I18n, vos fichiers SASS et vos vues (nous ne les utiliserons pas ici, l'objectif est de construire une petite API qui sera utilisable avec React ou Vue.js)
- **routes**: ce qui permettra à Laravel de savoir quoi faire et quand. Selon l'utilisation d'une API, des commandes ou d'une application monolithique. Il contient également un fichier pour gérer les évènements broadcasted de votre application.
- **storage**: sert entre autres à stocker les logs de votre application.
- **vendor**: toutes les dépendances y seront stockées.

Dans les migrations, vous trouverez déjà une table pour les utilisateurs et la réinitialisation des mots de passe. Malheureusement, c'est très long d'expliquer ceci dans un article. Vous pouvez néanmoins lire la documentation officielle pour gérer les utilisateurs avec une API REST sur la documentation de Laravel avec Passport[1].
Créons notre premier modèle et une migration pour gérer nos tâches à faire de la semaine :

```
php artisan make:model Todo -mrf
```

Petit coup d'œil aux paramètres de cette ligne de commande :

- **php artisan** : l'outil utilisé pour générer des fichiers dans votre application. N'hésitez pas à consulter tout ce qu'il peut faire en écrivant `php artisan` sur votre terminal ou **php artisan make:model –help** dans notre cas.
- **make:model** : permet la création d'un modèle dans votre application.
- **-m** : création d'une migration en même temps que le modèle.
- **-r** : crée le contrôleur en tant que ressource.
- **-f** : création de la factory pour le modèle.

# Abonnez-vous à Programmez! Abonnez-vous à Programmez! Abonne

[Programmez!] Le magazine des développeurs

## Nos classiques

| | |
|---|---|
| 1 an<br>11 numéros | 49€* |
| 2 ans<br>22 numéros | 79€* |
| Etudiant<br>1 an - 11 numéros | 39€* |

** Tarifs France métropolitaine*

## Abonnement numérique

| | |
|---|---|
| PDF<br>1 an - 11 numéros | 35€ |
| Option : accès aux archives | 15€ |

Souscription uniquement sur
www.programmez.com


**Toutes nos offres sur www.programmez.com**

# Oui, je m'abonne

ABONNEMENT à retourner avec votre règlement à :
Service Abonnements PROGRAMMEZ, 4 Rue de Mouchy, 60438 Noailles Cedex.

☐ **Abonnement 1 an :** 49 €
☐ **Abonnement 2 ans :** 79 €
☐ **Abonnement 1 an Etudiant :** 39 €
Photocopie de la carte d'étudiant à joindre

**PROG 237**
Valable jusqu'au 28 février 2020

☐ Mme ☐ M. Entreprise : Fonction :

Prénom : Nom :

Adresse :

Code postal : Ville :

email indispensable pour l'envoi d'informations relatives à votre abonnement

E-mail : @

☐ Je joins mon règlement par chèque à l'ordre de Programmez !
☐ Je souhaite régler à réception de facture

*** Tarifs France métropolitaine**

De retour sur votre IDE, nous avons maintenant besoin d'ajouter des champs dans la base de données. Rendez-vous dans database/migrations/xxxx_xx_xx_xxxxxx_create_todos_table.php.
L'artisan vous ayant prémâché le travail, il y a déjà deux attributs dans la migration: l'ID en identifiant unique et auto-incrémenté et created_at, ains que updated_at. Ces deux derniers peuvent être null dans votre base de données.
Il y a déjà deux méthodes dans la classe CreateTodoTable: up et down. Up sera la méthode exécutée lorsque la migration sera effectuée et down lorsqu'elle sera annulée.
Ajoutons quelques champs :

- **Task** en string pour le nom de la tâche à réaliser.
- **Status** en enum pour avoir son statut à tout moment.

Vous pouvez trouver la liste des options disponibles pour les migrations sur la documentation officielle[2].
Mais cette énumération n'existe pas encore. Créons donc le fichier config/enums.php. Dedans, un petit tableau est suffisant avec les informations que status peut avoir.

```
<?php

return [
  "status" => [
    "todo" => "To do",
    "blocked" => "Blocked",
    "doing" => "Doing",
    "done" => "Done",
  ],
];
```

Continuons la configuration de notre table. Nous devrons utiliser les clés de l'énumération status. La clé: array_keys.
La méthode up ressemblera donc à ceci :

```
public function up()
{
  Schema::create('todos', function (Blueprint $table) {
    $table->bigIncrements(' id');
    $table->string('task');
    $table->enum('status', array_keys(\Config::get('enums.status')));
    $table->timestamps();
  });
}
```

Vous pouvez à présent exécuter les migrations avec php artisan migrate.
Si vous vérifiez votre table todos, les colonnes suivantes apparaîtront: id, task, status, created_at et updated_at. Petit coup d'œil sur-le-champ status que nous avons créés en tant qu'énumération: il s'agit d'un VARCHAR. Pas de soucis, c'est Laravel qui s'occupe de tout ça. Il vérifiera si l'entrée du modèle correspond bien aux entrées disponibles pour le champ.
Maintenant, passons à la création du modèle en lui-même. Rendez-vous sur app/Todo.php.
Le modèle est vide, c'est tout à fait normal. Avant de commencer, il faut savoir que Laravel a une protection sur les champs. Deux variables sont présentes dans sa classe mère $fillable et $guarded.
La variable $fillable permet de lui dire explicitement quels champs sont éditables et inversement pour $guarded. Par exemple, l'ID ne peut pas être éditable. Si vous souhaitez accepter ou refuser la totalité des champs, un petit ['*'] suffira, mais c'est très largement déconseillé. Ils sont également utilisés lors de l'enregistrement de masse d'un tableau. Configurons donc les champs éditables:

```
class Todo extends Model
{
  protected $fillable = ["task", "status"];
}
```

À savoir que les champs created_at, updated_at sont automatiquement modifiables. L'ID ne l'est pas.
Notre modèle est enfin prêt

## Passons au contrôleur

Avant de faire quoi que ce soit dans le contrôleur, configurons la route. Dans routes/api.php. Nous allons faire les choses manuellement pour bien comprendre la situation. Avec des routes classiques, Laravel ne détecte pas les méthodes PUT et PATCH, car il est dans un contexte monolithique, où on précise, dans les templates de l'application, la méthode à employer grâce à @method('PUT'). Nous verrons juste après comment faire pour les supporter dans notre API. Nous allons juste écrire les routes pour récupérer les Todos, une Todo et en ajouter une ce qui est suffisant pour comprendre un truc génial de Laravel.
Dans routes/api.php, ajoutons ceci:

```
Route::get('/todos', 'TodoController@index');
Route::get('/todos/{todo}', 'TodoController@show');
Route::post('/todos', 'TodoController@store');
```

Nous appelons les méthodes get et post de Route qui sont des méthodes statiques. Le premier paramètre est la route et le second le contrôleur et la méthode de celui-ci séparé par une arobase (@). Sur la deuxième ligne, nous avons écrit **{todo}**, mais pourquoi ? Nous enverrons l'ID de la Todo souhaitée, Laravel ira chercher l'information en base de données tout seul et celle-ci sera disponible avec le modèle Todo que nous avons créé plus tôt et qui sera un paramètre des méthodes du contrôleur.
Sur le terminal, tapons:

```
php artisan route:list
```

Il va nous lister les routes qui sont configurées et accessibles. Vous remarquerez qu'il y a des middlewares qui sont déjà présents. Ce sont des routes qui exécutent un middleware avant que le contrôleur ne soit appelé. Les utiliser dans les routes n'est qu'une des nombreuses méthodes pour exécuter un middleware (ou plusieurs). Avec Laravel, ils permettent de modifier les requêtes et uniquement celles-ci. Voici un petit échantillon des méthodes disponibles pour exécuter des middlewares avec Laravel :

- Dans le constructeur d'un contrôleur avec $this→middleware('name')
- Dans les routes après leur déclaration grâce à Route::get('example', 'MyController@index')→middleware('auth')
- Exécuter une liste de middlewares par groupe de routes (web, api)
- Exécuter un middleware dans tous les cas.

Si vous désirez en apprendre davantage, je vous conseille d'aller lire

les documentations concernant les middlewares[3] et les contrôleurs .
Dans notre fichier todo/routes/api.php, supprimons les trois lignes écrites plus hauts pour les remplacer par une seule

```
Route::resource('todo', 'TodoController');
```

Encore une fois, pas d'étalement, une seule méthode est montrée ici, si je devrais expliquer les différentes manières de faire des routes, il faudrait un dossier complet. Comme d'habitude, rendez-vous sur la documentation concernant le routing .
Faites un

```
php artisan route:list
```

dans le terminal. Il y a toutes les routes présentes pour un CRUD avec les bonnes méthodes. 2
Direction app/Http/Controllers/TodoController.php. Dedans il y a déjà des méthodes. Explications :

- **index**: permet de lister toutes les données.
- **store**: créer une Todo, en POST.
- **show**: montrer une Todo uniquement.
- **update**: mise à jour d'une Todo.
- **destroy**: détruit la Todo.

Vous pouvez supprimer les autres méthodes, elles ne sont pas utiles dans notre contexte. Commençons par index. Renvoyons toutes les tâches qui sont existantes. Écrivons sobrement :

```
public function index()
{
  return Todo::all();
}
```

Faisons tourner notre application sur un terminal grâce à php artisan serve. Ensuite, lançons une requête pour récupérer toutes les Todos: curl localhost:8000/api/todos : c'est vide.
Chaque problème a sa solution: premièrement, ajoutons une Todo. Dans un terminal, entrez php artisan tinker. Cela ouvrira un interpréteur capable de communiquer avec votre application. Dans celui-ci, tapons juste ceci :

```
\App\Todo::create(['task' => 'Learn Laravel with Programmez!', 'status' => 'doing']) ;
```

Refaites une requête avec CURL sur l'endpoint et…

```
[{"id":"1","task":"Learn Laravel with Programmez!","status":"doing","created_at":"2019-11-09 17:03:12","updated_at":"2019-11-09 17:03:12"}]%
```

## Faisons simple pour ne pas trop coder

Mais nous, développeurs/euses, sommes flemmard(e)s. Nous allons donc en créer à la pelle pour des raisons de facilité. Vous vous rappelez les factories de tout à l'heure ?
Dans votre IDE, ouvrez le fichier database/factories/TodoFactory.php. La classe étant toute neuve, elle ressemble à ceci :

```
$factory->define(Todo::class, function (Faker $faker) {
  return [
    //
  ];
});
```

```
todo : zsh — Konsole
➜  todo php artisan route:list
+--------+-----------+----------------------+--------------+-----------------------------------------------------+--------------+
| Domain | Method    | URI                  | Name         | Action                                              | Middleware   |
+--------+-----------+----------------------+--------------+-----------------------------------------------------+--------------+
|        | GET|HEAD  | /                    |              | Closure                                             | web          |
|        | GET|HEAD  | api/todos            | todos.index  | App\Http\Controllers\TodoController@index           | api          |
|        | POST      | api/todos            | todos.store  | App\Http\Controllers\TodoController@store           | api          |
|        | GET|HEAD  | api/todos/create     | todos.create | App\Http\Controllers\TodoController@create          | api          |
|        | GET|HEAD  | api/todos/{todo}     | todos.show   | App\Http\Controllers\TodoController@show            | api          |
|        | PUT|PATCH | api/todos/{todo}     | todos.update | App\Http\Controllers\TodoController@update          | api          |
|        | DELETE    | api/todos/{todo}     | todos.destroy| App\Http\Controllers\TodoController@destroy         | api          |
|        | GET|HEAD  | api/todos/{todo}/edit| todos.edit   | App\Http\Controllers\TodoController@edit            | api          |
|        | GET|HEAD  | api/user             |              | Closure                                             | api,auth:api |
+--------+-----------+----------------------+--------------+-----------------------------------------------------+--------------+
➜  todo
```

2

Dans le tableau de retour, nous allons modifier les attributs qui serviront à créer quelques tâches que nous devons réaliser.

```
$factory->define(Todo::class, function (Faker $faker) {
  return [
    'task' => $faker->text(),
    'status' => 'todo',
  ];
});
```

Maintenant, nous avons besoin d'un seeder. Dans un terminal, entrez :

```
php artisan make:seeder TodoSeeder
```

Dans le dossier seeds en dessous du dossier factories, le nouveau seeder apparaît. Avant de le modifier, nous devons l'enregistrer dans /app/database/seeders/DatabaseSeeder.php. Dans la méthode run(), il y a déjà un seeder commenté. Nous devons juste ajouter le nôtre dedans :

```
public function run()
{
  // $this->call(UsersTableSeeder::class);
  $this->call(TodoSeeder::class);
}
```

Ajoutons à présent de quoi créer 20 tâches à réaliser dans le seed et ajoutons-les dans la base de données.

```
public function run()
{
  factory(\App\Todo::class, 20)->create();
}
```

Dans le terminal, tapez :

```
php artisan db:seed
```

Refaites une requête pour récupérer toutes les tâches et vous en verrez 21 apparaître. Les 20 dernières ont été générées toutes seules par la factory. Imaginez que vous ayez 100 tâches enregistrées. Souhaitez-vous vraiment renvoyer les 100 tâches d'un coup ? Probablement pas. Tout d'abord, il faut créer la ressource avec l'artisan.

```
php artisan make:resource TodoResource
```

Le nouveau fichier (la ressource) se situe dans /app/Http/Resources. Elle ne contient qu'une méthode toArray qui retourne la méthode parente toArray.

De retour dans le contrôleur, nous allons retourner la ressource pour faire les choses proprement.

```
public function index()
{
  return TodoResource::collection(Todo::paginate());
}
```

Petite particularité : on renvoie une collection. C'est-à-dire un ensemble de Todo. Celles-ci sont paginées par défaut par 15. Si vous souhaitez modifier le nombre de Todos renvoyées, il suffit d'y ajouter un paramètre (entier).
Si vous faites une nouvelle requête pour récupérer les tâches, vous y trouverez plusieurs objets JSON: data pour les tâches renvoyées, links pour la navigation à travers l'endpoint et meta pour les informations concernant la pagination.
Restons dans notre contrôleur.
Cette fois, nous allons modifier la méthode show qui permet d'envoyer les données d'une Todo et une seule. Remarquez le paramètre de la méthode, il n'y a que $id. Si nous restons avec ce paramètre, il faudrait rechercher les Todos enregistrées qui ont cet ID, comme ceci :

```
public function show($id)
{
  $todo = Todo::find($id);
  if(is_null($todo)) {
    return response()->json([], 404);
  }
  return new TodoResource($todo);
}
```

Il existe cependant des méthodes plus courtes, nous pouvons simplement utiliser findOrFail comme ceci :

```
public function show($id)
{
  $todo = Todo::findOrFail($id);
  return new TodoResource($todo);
}
```

D'abord, on va chercher la Todo ayant l'ID reçu en paramètre, si **Eloquent** (l'ORM utilisé par Laravel) ne le trouve pas, il renvoie null et nous retourne une 404, sinon la Todo.
Dans les routes, il y a ceci: /*api*/todos/{todo}/. Laravel va chercher automatiquement la Todo ayant cet ID. Ayant ça en tête, mettons à jour la méthode show :

```
public function show(Todo $todo)
{
  return new TodoResource($todo);
}
```

C'est comme ceci que Laravel fonctionne et c'est dans cette optique que nous devons développer. Les contrôleurs doivent rester les plus concis possible.

Dans la même optique, on va mettre à jour les méthodes update et destroy.
Cette dernière sera très simple: modification du paramètre, suppression de $todo et nous renvoie une 204.

```
public function destroy(Todo $todo)
{
  $todo->delete();
  return response()->json([], 204);
}
```

Pour update, pareil avec une petite subtilité: l'assignement de masse. Lorsque nous avons créé une Todo avec tinker auparavant, nous avons envoyé un tableau (pour rappel : \App\Todo::create(['task' => 'Learn Laravel with Programmez!' ;).
Dans une méthode, on a souvent mis la Todo comme paramètre. Dans le cadre de l'update, il y aura un autre paramètre $request qui est présent avant le paramètre Todo. Cette variable contiendra ce qui a été reçu par la requête JSON, y compris les nouvelles valeurs de la Todo.

```
public function update(Request $request, Todo $todo)
{
  $todo->update($request->all());
  return new TodoResource($todo);
}
```

Explications: $request→all() va renvoyer les attributs reçus par la requête, mettre à jour la $todo en accord avec les variables $fillable et $guarded et on retourne la Todo qui a été mise à jour.
Pour vérifier le fonctionnement de la mise à jour avec CURL:

```
curl --header "Content-Type: application/json" --request PUT --data '{"status": "done
"}' localhost:8000/api/todos/4
```

Laravel va vous retourner le nouvel état de l'objet après sa mise à jour.

## Conclusion

Il y a plein d'autres choses à faire avec Laravel que nous n'avons pas eu le temps de voir. La gestion des clés étrangères, allez plus loin avec la philosophie du framework pour gérer les relations, les commandes, les jobs, les tests, etc. Encore plus pour parler de l'écosystème comme Forge, Horizon, Nova, etc. Il faudrait un numéro spécial pour faire découvrir l'étendue phénoménale de Laravel. Ses librairies sont d'excellentes qualités, la communauté est énorme, même si elle est surtout anglophone.

### Liens

(1) https://laravel.com/docs/master/passport
(2) https://laravel.com/docs/master/migrations
(3) https://laravel.com/docs/master/middleware
(4) https://laravel.com/docs/master/controllers
(5) https://laravel.com/docs/master/routing

**Thierry LERICHE**
*Architecte et tech lead*
@ThierryLeriche

# Les nouveautés de PHP 7.4

*La version 7.4 de PHP est disponible depuis fin novembre. Il s'agit d'une version qualifiée de mineure, mais qui mérite d'être connue. Elle améliore (encore) les performances et propose quelques nouveautés réellement intéressantes, notamment au niveau de la lisibilité du code. Petite review de notre part !*

## Installation

L'installation et/ou la mise à jour de cette version se font de manière classique sous Linux. Vous aurez besoin de fpm et de vos dépendances habituelles (ici notées *xxx*) comme *curl*, *gd*, *mysql*, etc.

```
apt-get update
apt-get upgrade

apt-get install php7.4
apt-get install php7.4-fpm
apt-get install php7.4-xxx
```

**Note**

*Vous devrez probablement mettre à jour votre système. Et, comme souvent,* **libc6** *devrait vous faire perdre quelques cheveux.*

## Propriétés typées

Il est désormais possible de spécifier le type des propriétés des classes, y compris avec vos propres types.

```
class Dog {
  public string $id;
  public string $name;
  public int $age;
  public array $colors;
  public Person $owner;
```

Si on tente d'affecter une valeur qui ne correspond pas, par exemple la *string* « *hello* » dans un *int*, alors on obtient une erreur (ici « *$age must be int* »).

```
class Dog {
  public int $age;
  ...

$dog = new Dog();
$dog->age = 'hello'; // Fatal error
```

Toutefois, ça passe si la valeur est transformable (compatible) vers le type spécifié.

```
class Dog {
  public int $age;
  ...

$dog = new Dog();
$dog->age = '12'; // ça passe...
```

Il est bien entendu possible de changer ce comportement par défaut en le déclarant plus strict.

```
declare(strict_types=1);
...

class Dog {
  public int $age;
  ...

$dog = new Dog();
$dog->age = '12'; // Fatal error
```

Et bien entendu, il est possible de préciser une valeur à la déclaration ou dans le constructeur.

```
class Dog {
  public int $age = 12;
  ...
```

Par contre, il n'y a pas d'initialisation automatique des propriétés, comme dans d'autres langages typés. Il faut donc le faire manuellement avant de les utiliser. Dans l'exemple, on obtient l'erreur « *$age must not be accessed before initialization* ». Et marquer la valeur comme *nullable* (avec le caractère « ? »), ne change rien…

```
class Dog {
  public int $age;
  ...

$dog = new Dog();
$isAdult = 18 <= $dog->age; // Fatal error
```

## Covariance

Imaginons la classe *Animal* et la classe *Dog*.

```
class Animal {
  public string $name;
  ...

class Dog extends Animal {
  ...
```

Imaginons maintenant la classe *Person* et la classe *DogLover*, qui en étend. Et ajoutons la fonction *favoriteAnimal* qui renvoie un *Animal*. Jusqu'à PHP 7.3, la fonction *favoriteAnimal* de *DogLover*

devait avoir une signature équivalente, même si le contenu était spécialisé.

```
lass Person {
  public function favoriteAnimal(string $name) : Animal {
    $animal = new Animal();
    $animal->name = $name;
    return $animal;

class DogLover extends Person {
  public function favoriteAnimal(string $name) : Animal {
    $dog = new Dog();
    $dog->name = $name;
    return $dog;

$lover = new DogLover();
$favorite = $lover->favoriteAnimal('Milou');
```

Mais partant du principe que *Dog* étend *Animal*, on aimerait spécifier un prototype plus spécialisé (= plus précis) pour la fonction, ce qui devient possible en 7.4. Cela provoquait l'erreur de déclaration « *DogLover::favoriteAnimal(string $name): Dog must be compatible with Person::favoriteAnimal(string $name)* » en version 7.3.

```
class Person {
  public function favoriteAnimal(string $name) : Animal {
    ...

class DogLover extends Person {
  public function favoriteAnimal(string $name) : Dog {
    ...
```

Cette spécialisation n'est pas (encore) possible pour les paramètres des fonctions. L'exemple suivant déclenche un warning au niveau de la déclaration « *DogLover::foo(Dog $param) should be compatible with Person::foo(Animal $param)* ».

```
class Person {
  public function foo(Animal $param) {
    ...

class DogLover extends Person {
  public function foo(Dog $param) {
    ...
```

En revanche, il est possible d'écrire une *contravariance*, avec un typage plus large des paramètres.

```
class Person {
  public function bla(Dog $param) {
    ...

class DogLover extends Person {
  public function bla(Animal $param) {
    ...
```

## Opérateur Spread

La concaténation de tableau était assez laborieuse jusqu'en version 7.3. Le plus facile était encore d'utiliser la fonction *array_merge(1)*, qui ne simplifie pas la lecture.

```
$tab1 = [1, 2, 3];
$tab2 = [4, 5];
$all = array_merge($tab1, $tab2, [6, 7, 8]);
```

PHP 7.4 propose désormais une syntaxe similaire à celle de JavaScript.

```
$all = [...$tab1, ...$tab2, 6, 7, 8];
```

Il est également possible de combiner la syntaxe Spread avec la syntaxe Generator.

```
$tab1 = [1, 2, 3];
$tab2 = [4, 5];

function generator() {
    for ($i = 6; $i < 10; $i++) {
        yield $i;

$all = [...$tab1, ...$tab2, ...generator()];
```

## Opérateur d'assignation Coalesce Null

Les versions précédentes permettaient de gérer l'éventuelle nullité d'une variable, en lui passant une valeur de remplacement par défaut.

```
$a = 123;

$a = $a ?? 987;         // a = 123
$b = $b ?? 'hello';     // b = hello
```

Cette syntaxe était toutefois un peu lourde. Elle a été simplifiée dans PHP 7.4

```
$a = 123;

$a ??= 987;         // a = 123
$b ??= 'hello';     // b = hello
```

**Note**

*l'opérateur Spread devrait apporter un gain important au niveau des performances. À vérifier dans le temps…*

## Séparateur numérique

Pour plus de lisibilité, les valeurs numériques peuvent désormais inclure le caractère *underscore*. Il vous servira habituellement à identifier les milliers, mais on peut le placer n'importe où. Ce n'est pas un ajout fondamental, mais c'est bien pratique.

```
$f = 6.674_083e-11; // flotant
$e = 299_792_458;   // entier
$n = 1_2_3_4_5;     // n'importe où
$h = 0xCAFE_F00D;   // hexa
$b = 0b0101_1111;   // binaire
```

## Références faibles

Les références faibles permettent de retenir une référence sans empêcher l'objet d'être détruit.

Pour l'exemple, considérons les classes A et B, avec une dépendance de B vers A. La destruction (*unset*) de A n'est pas totalement possible, car la référence est utilisée dans B.

```php
class A {
  public string $name;
}

class B {
  public A $a;
}

$a = new A();
$a->name = 'foo';

$b = new B();
$b->a = $a;

var_dump($a);  // Contenu de $a
unset($a);     // Pas possible car utilisé par $b
var_dump($a);  // Error : Undefined variable
var_dump($b);  // $a existe encore dans $b
```

Pour résoudre ce souci, PHP 7.4 propose d'utiliser une référence faible. On la crée grâce à *WeakReference::create* à qui on passe l'objet concerné, et on la récupère avec *get* tout simplement. Pour que ça fonctionne, mais aussi pour des raisons de clarté, le type doit être indiqué dans la classe B.

```php
class B {
  public WeakReference $a;
}

$b = new B();
$weakRef = WeakReference::create($a);
$b->a = $weakRef;

var_dump($a);   // Contenu de $a
var_dump($b->a->get());  // Get de $a via sa ref
unset($a);     // Pas possible car utilisé par $b
var_dump($a);  // Error : Undefined variable
var_dump($b);  // $a n'existe plus dans $b
```

Ça devrait particulièrement vous intéresser, en particulier si vous faites de l'injection de dépendances ou si vous avez des fuites/problèmes de mémoire.

## Lambda

Jusqu'ici, il était possible d'écrire (et de passer) des fonctions, par exemple, pour filtrer les éléments impairs d'un tableau.

```php
$tab = [1, 2, 3, 4, 5, 6, 7, 8, 9];

$odd = array_filter($tab, function($item) {
  return $item % 2 !== 0;
});
```

Cette syntaxe était lourde et pénalisait la lisibilité. PHP 7.4 ajoute les fonctions flèches, inspirées d'autres langages. Les accolades et le *return* ne sont alors plus nécessaires. Ils deviennent d'ailleurs interdits. Il est encore nécessaire d'écrire *fn(2)*, mais c'est déjà bien plus simple…

```php
$tab = [1, 2, 3, 4, 5, 6, 7, 8, 9];

$odd = array_filter( $tab, fn($item) => $item % 2 !== 0 );
```

Et bien entendu, la fonction flèche peut être définie séparément, et réutilisée.

```php
$tab = [1, 2, 3, 4, 5, 6, 7, 8, 9];

$oddFunction = fn($item) => $item % 2 !== 0;
$odd = array_filter( $tab, $oddFunction );
```

Cerise sur le gâteau, les fonctions flèche ont accès aux variables extérieures(3) sans devoir utiliser le mot de clé *use*.

```php
$DEUX = 2;
$oddFunction1 = function ($item) use ($DEUX) {
  return $item % $DEUX !== 0;
};

$oddFunction2 = fn ($item) => $item % $DEUX !== 0;
```

## En vrac

Il est désormais possible de lancer des exceptions depuis *__toString* alors que ça provoquait une erreur fatale dans les versions précédentes. Les erreurs récupérables dans les conversions sont maintenant des exceptions *Error*.

Le support pour le préchargement (OPcache) de code a été ajouté. On en parlait dans un précédent numéro du magazine.

Les opérateurs ternaires imbriqués doivent désormais utiliser des parenthèses pour indiquer l'ordre des opérations. Cela a pour objectif de rétablir le sens de lecture. Personnellement, je pense que c'était une mauvaise pratique. De même les concaténations multi-types doivent également être avec des parenthèses.

L'utilisation de *array_key_exists* sur des objets devient *deprecated*, au profit de *isset* et *property_exists*.

L'algo CRC32C (variante de CRC32, polynôme de Castagnoli) vient enrichir le *hash*. C'est pratique quand on travaille avec des certains systèmes de stockage (iSCSI, SCTP, Btrfs, ext4…).

Les chaînes de connexion PDO ont été simplifiées, notamment pour indiquer le *user* et le *password*.

## Conclusion

Un certain nombre d'éléments entraînent une incompatibilité ascendance et certaines fonctionnalités deviennent obsolètes, mais il n'y a rien d'insurmontable. Et pour savoir ce qui change, le mieux est encore de se référer au guide de migration(4).

Alors, faut-il migrer vers 7.4 ? Comme souvent, ce sera une réponse de normand… Ce n'est pas du tout obligatoire. Toutefois, si vous en avez l'occasion, vous gagnerez en performances et votre code deviendra (beaucoup) plus lisible. Il serait alors dommage de s'en passer.

---

*(1) Les fonctions array_merge et array_merge_recursive peuvent désormais être appelées sans argument, et retourner un tableau vide le cas échéant.*

*(2) fn est maintenant un mot-clé réservé et ne peut plus être utilisé pour une fonction ou un nom de classe.*

*(3) Qui ne doivent pas être mutées, ce qui est plutôt une bonne chose*

*(4) Guide de migration :* https://www.php.net/manual/en/migration74.php

# Back to basics : Systemd

**Marina Rouillé**
*Développeuse web chez* **Anaya.**

***Systemd est un système qui permet de gérer le démarrage d'un noyau Linux et d'automatiser un grand nombre de tâches : démarrage d'applications au démarrage du serveur, planification des traitements récurrents…***

L'approche peut paraître simple mais elle s'avère très puissante. Pour la planification, je trouve que c'est beaucoup plus propre et lisible que cron. Et le bonus c'est que tout est loggé et facilement accessible avec la commande *journalctl*. *systemctl* est la commande qui permet d'utiliser systemd.

**N.B.** *Les instructions de l'article valent pour une distribution Linux RHEL 7.0. Tous les fichiers sont créés dans le répertoire /usr/lib/systemd/system, on ne touche pas à /etc, on le laisse dédié à l'administration système.*

## Lancement d'un service au démarrage d'une instance de serveur

Alors pour démarrer automatiquement un service au démarrage de l'instance, on va déclarer le service, l'activer et voilà !

### La déclaration

On doit créer un fichier de description du service dans le répertoire */usr/lib/systemd/system* (on réserve */etc/systemd/system* pour l'administration système). Le fichier doit être nommé au format *nom.service*. Dans cet exemple nous le nommons *api-poc.service*. Au minimum voilà à quoi il va ressembler :

```
[Unit]
Description=Démarrage de l'application POC

[Service]
ExecStart=/bin/bash /applications/chemin_tortueux/runApiPoc.sh

[Install]
WantedBy=multi-user.target
```

**[Unit]** permet de décrire le service
**[Service]** contient notamment la commande associée au service défini dans ExecStart C'est dans cette section qu'on pourra enrichir la définition avec le contexte(user et group avec lequel lancer le service), le type de service (par défaut il est "simple"), le chargement d'une configuration pour fournir des variables d'environnement… On peut également définir des instructions à exécuter avant ou après ExecStart, des instructions spécifiques à l'arrêt de l'application...
**[Install]** permet de définir le niveau d'exécution du service, c'est-à-dire à quel moment vous souhaitez que votre service soit lancé et donc aussi à partir de quel moment les services eux-même nécessaires au vôtre sont disponibles. Systemd définit ce moment via des targets, le multi-user correspond aux runlevels de 2 à 4. Pour une définition des runlevels voir http://www.linfo.org/ runlevel_def.html

### L'activation

Et maintenant pour que le service soit démarré à chaque démarrage du serveur automatiquement :

```
invite_commande > sudo systemctl enable api-poc
Created symlink /etc/systemd/system/multi-user.target.wants/api-poc.service -> /usr/lib/systemd/system/api-poc.service.
```

Et voilà, votre service, ici notre API, sera démarré automatiquement à chaque démarrage du serveur.

## Planifier une tâche

Et pour exécuter une tâche planifiée ? Très simple aussi ! On va définir un service et lui adjoindre un timer. On définit un service qu'on voudra planifier, par exemple un traitement journalier, *batch-poc.service* :

```
[Unit]
Description=Traitement à appliquer quotidiennement sur la base de données

[Service]
ExecStart=/bin/bash /applications/chemin_tortueux/runBatchPoc.sh
```

Pas la peine de préciser la section [Install] : son exécution ne sera plus déclenchée automatiquement en fonction du runlevel du serveur mais quand le timer l'appellera.

### La déclaration du timer

On créer le timer associé qui doit porter le même nom mais l'extension .timer, *batch-poc.timer* :

```
[Unit]
Description=Planification du traitement sur la base de données

[Timer]
OnCalendar=*-*-* 03:00:00

[Install]
WantedBy=timers.target
```

Ici au lieu de la section [Service], on renseigne une partie [Timer] pour définir au minimum le moment auquel on veut que le service soit exécuté, ici on demande une exécution tous les jours à 3h00. La syntaxe est proche de celle du cron et je trouve un peu moins cryptique! Cette doc est pas mal pour la syntaxe : https://www.certdepot.net/rhel7-use-systemd-timers/

### L'activation

Au lieu d'activer le service, on active le timer :

```
invite_commande > sudo systemctl enable batch-poc.timer
Created symlink /etc/systemd/system/timers.target.wants/batch-poc.timer -> /usr/lib/systemd/system/batch-poc.timer.
```

Et on le démarre :

```
invite_commande > sudo systemctl start batch-poc.timer
```

On peut maintenant le voir dans la liste des timers :

```
invite_commande > sudo systemctl list-timers
NEXT                         LEFT    LAST  PASSED    UNIT           ACTIVATES
Sat 2019-12-13 03:00:00 UTC  8h left n/a   n/a       batch-poc.timer batch-poc.service
```

La commande permet de vérifier notamment la dernière fois que le timer a été exécuté et dans combien de temps il sera exécuté à nouveau.

## Au-delà de la base

Voilà le minimum pour se servir de systemd, ensuite vous pouvez enrichir vos services et vos timers avec toutes les possibilités offertes, parmi lesquelles :

- Préciser le moment où le service doit être démarré en utilisant "After" avec la liste de processus qui doivent être montés avant le démarrage du nôtre et/ou "Before" pour l'inverse !
- Préciser avec "ExecStop" la façon dont notre service doit être arrêté
- Exécuter le service avec un "Group" et un "User"
- Préciser le répertoire dans lequel le service doit être exécuté avec "WorkingDirectory"
- Définir un fichier à utiliser dans "EnvironnementFile" pour positionner les variables d'environnement à charger pour le service
- Définir un délai entre le déclenchement d'un timer et le démarrage du service qu'il appelle avec "OnActiveSec"

Il y a plein de documentations en ligne, testez !

Valentin Michalak
*Ingénieur Logiciel @ SOAT*

# Kotlin pour les développeurs Java

*A moins d'avoir vécu dans une grotte ses deux dernières années vous avez sûrement remarqué Kotlin, le langage de Jetbrains venu faire face au colosse Java. Depuis l'annonce de ce dernier comme langage officiel d'Android à la Google I/O 2017, Kotlin prend rapidement des parts de marché au point qu'il devienne le langage avec la plus grosse croissance sur GitHub(1) !*

(1) source: https://www.programmez.com/actualites/octoverse-report-de-github-les-principaux-langages-de-programmation-en-2018-28260

La popularité croissante de Kotlin lui a permis de s'émanciper d'Android. Les développeurs back ont commencé à s'y intéresser de près notamment depuis qu'il est officiellement supporté par Spring. Même si cela reste anecdotique pour le moment, il est aussi possible de sortir de la JVM et de compiler notre code Kotlin en JavaScript ou même sur iOS.

## Une petite page d'histoire

Jetbrains est une entreprise éditant des logiciels à destination des développeurs, le plus connu étant IntelliJ. Les équipes de Jetbrains sont habituées à utiliser Java pour développer leurs applications, mais plus le temps passe plus ils se rendent compte que le langage ne correspond plus forcément entièrement à leurs besoins. Après quelques recherches, aucun langage ne répondait à leurs besoins, l'idée de créer un nouveau langage germe.

Kotlin commence donc à se dessiner autour de plusieurs axes centraux. Le premier est de conserver une interopérabilité avec le code Java existant, les classes Kotlin et Java doivent pouvoir s'utiliser de manière totalement transparente. Le second est de réussir à se sortir des contraintes Java, d'avoir un code concis tout en restant simple à lire et à écrire. Le troisième est d'être sûr, fini les Null Pointer Exceptions car on a oublié de tester le null, fini de mettre final devant toutes les variables dont le contenu doit être immutable, etc.

## La syntaxe

### Variables (val / var)

En Kotlin, comme en Java, la question de la mutabilité d'une variable est importante pour des questions de maintenabilité et de sécurité. Lorsque l'on écrit une nouvelle variable, nous devons évaluer si son contenu va changer dans le temps ou pas. Il existe deux mots clés pour créer une variable en Kotlin : val si cette dernière est immutable, var si, à l'inverse, elle est mutable.

| Java | Kotlin |
|---|---|
| `var surname = "Valentin";`<br>`String surname = "Valentin";` | `var surname = "Valentin"`<br>`var surname: String = "Valentin"` |
| `final var surname = "Valentin";`<br>`final String surname = "Valentin";` | `val surname = "Valentin"`<br>`val surname: String = "Valentin"` |

### Egalités (== / ===)

En Java, une erreur commune est d'utiliser == (qui permet de vérifier que deux variables utilisent la même référence) à la place de `equals()` (qui permet de vérifier si deux valeurs sont les mêmes).

Pour éviter d'avoir ce problème en Kotlin, le == et le equals permettent tous les deux de comparer les valeurs et non pas les références. Pour les références, on ajoute un troisième égal : ===.

| Java | Kotlin |
|---|---|
| `final MyObject a = MyObject("a");`<br>`final MyObject b = MyObject("b");`<br><br>`a.equals(b);` | `val a = MyObject("a")`<br>`val b = MyObject("b")`<br><br>`a == b` |
| `final MyObject a = MyObject("a");`<br>`final MyObject b = MyObject("b");`<br><br>`a == b;` | `val a = MyObject("a")`<br>`val b = MyObject("b")`<br><br>`a === b` |

### Null Safety (reprendre la refcard)

Les nulls ne sont plus qu'une histoire ancienne pour les développeurs Kotlin, enfin presque. Par défaut en Kotlin, les types ne sont pas nullables !

Mais parfois il peut être intéressant ou utile d'avoir un type nullable (par exemple lorsque l'on récupère des données depuis une API ou lorsque l'on communique avec du code Java qui lui a des types nullables). Pour cela, Kotlin vous permet de rendre n'importe quel type nullable en ajoutant un `?` à la fin du type. Par exemple, un `String` s'il est nullable devient un `String?`

Lorsque l'on utilise un type nullable, on rentre un peu en conflit avec le compilateur qui a chaque utilisation de la variable va vous demander de vérifier si cette dernière est nulle, mais pas de panique, le langage va encore une fois vous aider à vous en sortir.

#### Opérateur d'Elvis

```
val result = a ?: b
```

L'opérateur d'Elvis est une bonne solution dans les cas où l'on veut remplacer `null` par une valeur par défaut. Dans l'exemple de code ci-dessus, lorsque a est `null` il est remplacé par b.

#### Safe Cast

```
val a: String? = "coucou"
val result = a?.length
```

Le Safe Cast permet d'accéder aux valeurs d'un type nullable comme s'il ne l'était pas, mais avec une particularité. Dans notre exemple ci-dessus, result est un `Int?` Tout simplement, car si a est `null` nous n'aurons pas une `Null Pointer Exception`, mais result sera à son tour `null`.

### L'opérateur !!

```
val a: String?= "coucou"
val b: String = a!!
```

L'opérateur `!!` est un opérateur à utiliser avec grande parcimonie. Lorsque vous utilisez l'opérateur `!!` Vous dites au compilateur *"Fait moi confiance, je sais que ce n'est pas `null`, je gère"*. Après si vous vous loupez, c'est votre problème.

## String Templates

La concaténation de `String` avec l'utilisation du `+` peut parfois être un peu laborieuse (surtout lorsque l'on a beaucoup de string). En Kotlin, une façon alternative existe avec l'utilisation du `$` devant une variable pour ajouter une `String` dans une autre `String`.

| Java | Kotlin |
|---|---|
| `final String text = "Mon prenom est" + surname + ".";` | `val text = "Mon prenom est $surname."` |

## Types Primitifs

En Kotlin, les types primitifs n'existent pas, tout est objet. Du coup pas de problématiques de performances liées au boxing et à l'unboxing.

## Tableaux (Arrays "Objects Generics")

En Kotlin, les tableaux sont représentés par des objets de type `Array<T>`.

## Déclaration de fonctions

En Kotlin, il est possible d'inliner la déclaration d'une fonction :

```
fun add(a: Int, b: Int) {
  return a + b
}
```

Devient alors :

```
fun add(a: Int, b: Int) = a + b
```

## Déclaration de classe

```
class User (
  surname: String,
  val lastName: String,
  private val age: Float
) {
   val surnameCapital = surname.capitalize()
}
```

Dans les bonnes pratiques Java, un constructeur ne doit pas effectuer de tâche lourde, au mieux il fait set des variables à l'intérieur de l'objet. En Kotlin, le constructeur est dès la première ligne de la construction de la classe entre les parenthèses. Le constructeur principal ne permet de n'assigner que des variables.
Dans notre exemple de constructeur de class ci-dessus, nous pouvons remarquer trois types de variables :

- La première `surname: String` permet d'exposer une variable pendant la phase d'initialisation de l'objet, mais disparaît ensuite. Elle peut être utile par exemple si elle permet d'initialiser une ou plusieurs variables à l'intérieur de l'objet.
- La seconde `val lastName: String` permet d'exposer une variable publiquement dans l'objet.
- La troisième private `val age: Float` permet d'exposer une variable privée à l'intérieur de l'objet.

## Data Classes

Les constructeurs principaux sont pratiques pour faire les bons vieux "POJO" ou models que l'on retrouve dans presque toutes les applications. Mais ils ne nous dispensent pas d'écrire les `getters`, `setters`, `equals`, `hashcodes` et `toString`. Vous l'avez deviné, en Kotlin encore une fois un mot clé est prévu à cet effet. Lorsque l'on rajoute le mot clé `data` devant le traditionnel `class`, la class devient une data class et le compilateur générera automatiquement toutes les méthodes classiques en se basant sur les variables du constructeur.

**Java**

```
public class LatLng {
  final float latitude;
  final float longitude;

  public LatLng(float latitude, float longitude) {
    this.latitude = latitude;
    this.longitude = longitude;
  }

  public float getLatitude() {
    return latitude;
  }

  public float getLongitude() {
    return longitude;
  }

  @Override
  public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    LatLng latLng = (LatLng) o;
    return Float.compare(latLng.latitude, latitude) == 0 &&
        Float.compare(latLng.longitude, longitude) == 0;
  }

  @Override
  public int hashCode() {
    return Objects.hash(latitude, longitude);
  }

  @Override
  public String toString() {
    return "LatLng{" +
        "latitude=" + latitude +
        ", longitude=" + longitude +
        '}';
  }
}
```

**Kotlin**

```
data class LatLng (
  val latitude: Float,
  val longitude: Float
)
```

## Objects (Singleton)

Bien que l'utilisation du Singleton soit assez discutée, il n'est pas rare de retrouver des singletons dans les projets. En Kotlin, pour

transformer une class en singleton, il suffit de remplacer le mot clé `class` par le mot clé `object`. Facile !

| Java | Kotlin |
|---|---|
| `public class MySingleton { private MySingleton() {} private static MySingleton INSTANCE; public static MySingleton getInstance() { if (INSTANCE == null) { INSTANCE = new MySingleton(); } return INSTANCE; } }` | `object MySingleton { }` |

## Méthodes d'extensions

Dans tous les gros projets Java nous retrouvons les class dites Utils comme StringUtils ou DateUtils. En Kotlin, au lieu de créer des classes pour cela il est possible d'ajouter des méthodes directement sur les classes d'origines. Dans l'exemple ci-dessous, on ajoute une méthode afin de générer un hash md5 directement sur l'objet `String`.

```
fun String.md5(): String {
  val md = MessageDigest.getInstance("MD5")
  return BigInteger(1, md.digest(toByteArray())).toString(16).padStart(32, '0')
}

fun main() {
  println("abc".md5())
}
```

### Infix

En Kotlin il est possible de créer des mots clés / opérateurs. Assez peu utile à moins de vouloir créer un DSL directement en Kotlin. Dans l'exemple ci-dessous, nous avons créé une méthode d'extension Infix pour vous donner une idée des possibilités.

```
infix fun Int.plus(x: Int) = this + x

fun main() {
  println(2 plus 2)
}
```

### Inline Function

Utiliser des lambdas impose un certain nombre de pénalités au runtime (allocation de mémoire, création d'objets intermédiaires, …). Pour éviter cela, il est possible d'utiliser des fonctions dites inlines. En ajoutant le mot clé `inline` devant une fonction, au moment de la compilation tous les appels à la fonction seront remplacés par le contenu de la fonction directement.

```
inline fun plus(a: Int, b: Int) = a + b
```

### When

Le switch-case est un opérateur très puissant, mais avec des restrictions d'usage et une syntaxe un peu particulière qui font que l'on le retrouve que très peu dans les projets finaux.
`When` vient donc remplacer le switch-case, avec une syntaxe beaucoup plus élégante que l'on peut voir ci-dessous.

| Java | Kotlin |
|---|---|
| `final bool answer = true; String text; switch(answer) { case true: text = "Vrai"; break; default: text = "Faux"; }` | `val answer = true val text = when (answer) { true -> "Vrai" false -> "Faux" }` |

Mais avec `when` en plus des constantes, nous pouvons utiliser des conditions comme ci-dessous.

```
var result = when(number) {
  0 -> "Invalide"
  1, 2 -> "Trop petit"
  3 -> "Correct"
  in 4..10 -> "Trop haut, mais acceptable"
  !in 100..Int.MAX_VALUE -> "Trop haut"
  else -> "Beaucoup trop haut"
}
```

Vous pouvez remarquer sur l'exemple ci-dessus en kotlin nous utilisons `else` au lieu d'utiliser le `default`.
Aussi, avec when, lorsque l'on utilise un test de type en guise de condition, le type est automatiquement casté !

```
when (x) {
  is Int -> print(x + 1)
  is String -> print(x.length + 1)
  is IntArray -> print(x.sum())
}
```

### Getter / Setters

En Kotlin, fini les méthodes `getMyValue` et `setMyValue` que l'on génère tous grâce à notre IDE favori. Avec Kotlin, il est possible de réécrire le fonctionnement du getter et du setter d'une variable directement sur celle-ci. Faire des méthodes de get / set n'est donc plus une best practice.

| Java | Kotlin |
|---|---|
| `int value = 0; void setValue(int value) { if(value < 0) { throw new IllegalArgumentException(); } this.value = value; }` | `var value = 0 set(v) { if(value < 0) { throw IllegalArgumentException(); } this.value = v }` |

### Surcharge des opérateurs

Sur Kotlin, il est possible de surcharger le fonctionnement des opérateurs pour toutes les classes. Plutôt pratique lorsque l'on travaille sur des opérations mathématiques.

```
data class Point(val x: Int, val y: Int)

operator fun Point.unaryMinus() = Point(-x, -y)

val point = Point(10, 20)

fun main() {
  println(-point)}
```

## Conclusion

Tester Kotlin, c'est l'adopter. J'espère que cet article vous donnera envie de tester le langage pour à votre tour comprendre pourquoi il est si apprécié de ses utilisateurs !

Anthony Giretti
*MVP, MCSD*
*Developer, Blogger, Speaker*
anthony.giretti@gmail.com
http://anthonygiretti.com

# Apprendre GIT en ligne de commande dans Visual Studio 2019 et GitHub

Partie 2

***Dans le précédent article, nous avons vu comment débuter avec GIT. La partie 1 évoquait : l'utilisation des bons outils pour travailler en ligne de commande, comment créer des dépôts, effectuer des changements sur les fichiers ainsi que la synchronisation des changements.***

Aujourd'hui, nous aborderons :

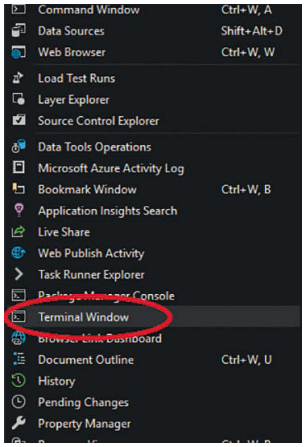
- Exclure des fichiers pour le suivi de version et vérifier l'historique des versions
- Chercher un commit et le comparer à un autre
- Revenir à un commit particulier

À noter que ce tutoriel (comme le précédent) est destiné aux débutants / intermédiaires, je n'aborderai pas toutes les options possibles de chaque commande GIT utilisées.

## Rappel des outils pour utiliser GIT en ligne de commande

Je vous ai proposé un terminal intégré à Visual Studio 2019 (et 2017) assez sympathique, il est nommé « Whack Whack » et il est disponible en téléchargement ici : https://marketplace.visualstudio.com/items?itemName=DanielGriffen.WhackWhackTerminal

Une fois installé vous devez activer le Windows Terminal dans le menu *View -> Other Windows*

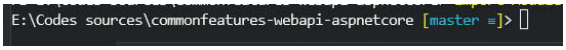



J'ai recommandé également une autre extension nommée **posh-git**, permettant d'indiquer en tout temps à côté du nom de la branche sur laquelle vous travaillez, les modifications courantes (fichiers modifiés, conflits, etc). Il est disponible en téléchargement ici : https://git-scm.com/book/en/v2/Appendix-A%3A-Git-in-Other-Environments-Git-in-PowerShell

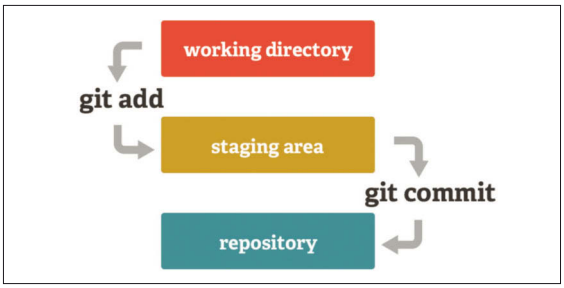
Si vous obtenez un message d'erreur indiquant que l'installation de **posh-git** est bloqué, car le script PowerShell n'est pas signé, veuillez exécuter cette commande qui lèvera les restrictions d'exécution de scripts PowerShell :

*Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass*

Si vous êtes déjà dans un repository GIT vous devriez obtenir quelque chose comme ceci :




**Rappel du fonctionnement général de GIT**




Avant d'aller plus loin dans cet article, je tiens à rappeler comment fonctionne GIT et quelles sont les différentes zones de travail. Dans cet article je vais utiliser le terme « répertoire de travail » qui correspond au « working directory » en rouge sur l'image. Une fois les modifications ajoutées à « l'index » qui est nommé « staging area », un commit va pousser les modifications dans ce que je nomme le « dépôt local », « repository » sur l'image.

## Exclure des fichiers du suivi de version et consulter l'historique de version

Exclusion de fichier

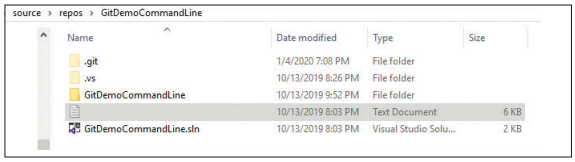
Avant de commencer à parcourir l'historique de version, nous allons voir comment exclure des fichiers du suivi de version. Nous avons deux scénarii :

- Nous savons dès la création du dépôt quels fichiers nous ne voudrons jamais inclure dans l'historique
- Nous découvrons pendant le cycle de développement que nous ne souhaitons plus suivre certains fichiers, et nous allons donc en arrêter le suivi et les ignorer totalement.

Dans les deux cas, nous devrons, comme je l'avais suggéré dans le précédent article, récupérer un fichier (.gitignore) dédié à l'environnement.NET téléchargeable ici : https://github.com/github/gitignore/blob/master/VisualStudio.gitignore

Si vous ne l'avez pas téléchargé, nous allons voir comment en créer un :

Allez à la racine de votre projet et créer un fichier nommé « .gitignore » (sans extension) et ajouter les types de fichiers que vous ne souhaitez pas suivre. Sous Windows le nom du fichier peut ne pas être visible comme ici :




Pour éditer le fichier, il vous suffit d'ajouter sur une nouvelle ligne le nom du fichier à ignorer, ou plusieurs fichiers commençant par n'importe quel nom « * » et se terminant par une extension de fichier particulière, à noter que le caractère « # » permet d'écrire un commentaire dans le fichier :


Une fois votre fichier créé et édité, il suffit de l'ajouter à l'index avec *git add .gitignore*, l'ajouter dans le dépôt local avec *git commit -m ''Ajout du fichier .gitignore ''*, puis le synchroniser avec le dépôt distant avec git push.

Voici les fichiers non désirés avant l'ajout du fichier « .gitignore » :

```
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .vs/
        GitDemoCommandLine/GitDemoCommandLine.csproj.user
        GitDemoCommandLine/Properties/
        GitDemoCommandLine/bin/
        GitDemoCommandLine/obj/
```

Et voici l'état de notre historique après ajout de ce fichier dans l'index puis dans le dépôt local, vous noterez que les fichiers non désirés ne sont plus dans le suivi de version, ils sont donc ignorés :

```
~\source\repos\GitDemoCommandLine [master = +6 ~0 -0 !]> git add .gitignore
warning: LF will be replaced by CRLF in .gitignore.
The file will have its original line endings in your working directory.
~\source\repos\GitDemoCommandLine [master = +1 ~0 -0 ~]> git commit -m "Ajout fichier .gitignore"
[master b91039f] Ajout fichier .gitignore
 1 file changed, 330 insertions(+)
 create mode 100644 .gitignore
~\source\repos\GitDemoCommandLine [master ↑1]> git push
Counting objects: 3, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 2.80 KiB | 2.80 MiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/AnthonyGiretti/GitDemoCommandLine.git
   ad8afec..b91039f  master -> master
~\source\repos\GitDemoCommandLine [master =]>
```

Si vous ne souhaitez pas avoir un fichier dans le suivi de version et que vous voulez qu'il soit ignoré, car vous souhaitez uniquement l'avoir dans votre copie locale, il suffit de l'ajouter dans le fichier « .gitignore »

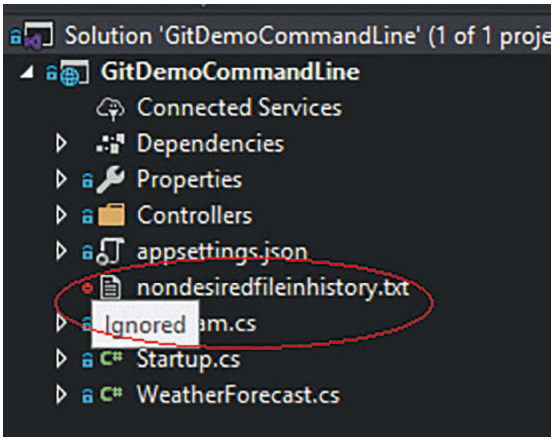
```
~\source\repos\GitDemoCommandLine [master = +1 ~0 -0 !]> git status
On branch master
Your branch is up to date with 'origin/master'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        GitDemoCommandLine/nondesiredfileinhistory.txt

nothing added to commit but untracked files present (use "git add" to track)
~\source\repos\GitDemoCommandLine [master = +1 ~0 -0 !]>
```

```
1  ## Ignore Visual Studio temporary files, build results, and
2  ## files generated by popular Visual Studio add-ons.
3  ##
4  ## Get latest from https://github.com/github/gitignore/blob/mas
5
6  # Specific files to ignore
7
8  GitDemoCommandLine/nondesiredfileinhistory.txt
9
```

Après modification de ce dernier, le fichier ignoré disparaît du suivi de version et le fichier « .gitignore » apparaît en fichier modifié que nous allons synchroniser à nouveau avec la même série de commandes vue plus haut lors de sa création.

```
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        GitDemoCommandLine/nondesiredfileinhistory.txt

nothing added to commit but untracked files present (use "git add" to track)
~\source\repos\GitDemoCommandLine [master = +1 ~0 -0 !]> git status
On branch master
Your branch is up to date with 'origin/master'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   .gitignore

no changes added to commit (use "git add" and/or "git commit -a")
~\source\repos\GitDemoCommandLine [master = +0 ~1 -0 !]>
```

Enfin une fois synchronisé, le fichier va apparaître avec la mention « ignoré » dans l'arborescence du projet




Si par hasard le fichier était déjà dans l'historique de version, exécutez la commande *git rm – cached nondesiredfileinhistory.txt* puis effectuez un commit et un push pour la synchronisation

Consulter l'historique de version

Il est possible à tout moment de consulter l'historique de version en ligne de commande tout comme il est possible de le voir sur Github. La commande *git log* va nous aider. Nous allons voir comment filtrer par date les logs, également filtrer pour obtenir les n derniers commit ou alors filtrer par auteur. Il existe d'autres options, mais celles si sont celles les plus utilisées.

Nous voulons voir par exemple les trois derniers commit, nous allons taper la commande *git log -3 :*

```
~\source\repos\GitDemoCommandLine [master =]> git log -3
commit 247e83eef24c802fe864dfa9580914ce26df3035 (HEAD -> master, origin/master, origin/HEAD)
Author: Anthony Giretti <anthony.giretti@gmail.com>
Date:   Sat Jan 4 20:08:20 2020 -0500

    ajout fichier non d<C3><A9>sir<C3><A9>

commit b91039ff6a6c6a19fa6a6fef9c041135051afacd
Author: Anthony Giretti <anthony.giretti@gmail.com>
Date:   Sat Jan 4 19:47:32 2020 -0500

    Ajout fichier .gitignore

commit ad8afec317c34319e118b3eb39ad5731874897cd
Author: Anthony Giretti <anthony.giretti@gmail.com>
Date:   Sat Jan 4 19:39:12 2020 -0500

    remove gitignore
~\source\repos\GitDemoCommandLine [master =]>
```

Nous voulons voir par exemple les commit d'Anthony Giretti, nous allons taper la commande *git log –author ''*Anthony Giretti*''* :

```
~\source\repos\GitDemoCommandLine [master ≡]> git log --author "Anthony Giretti"
commit 247e83eef24c802fe864dfa9580914ce26df3035 (HEAD -> master, origin/master, origin/HEAD)
Author: Anthony Giretti <anthony.giretti@gmail.com>
Date:   Sat Jan 4 20:08:20 2020 -0500

    ajout fichier non d<C3><A9>sir<C3><A9>

commit b91039ff6a6c6a19fa6a6fef9c041135051afacd
Author: Anthony Giretti <anthony.giretti@gmail.com>
Date:   Sat Jan 4 19:47:32 2020 -0500

    Ajout fichier .gitignore

commit ad8afec317c34319e118b3eb39ad5731874897cd
Author: Anthony Giretti <anthony.giretti@gmail.com>
Date:   Sat Jan 4 19:39:12 2020 -0500

    remove gitignore
```

Enfin nous voudrions voir par exemple les commits à partir d'une certaine date avec la commande *git log –since= ''2019-10-01''* :

```
~\source\repos\GitDemoCommandLine [master ≡]> git log --since="2019-10-01"
commit 247e83eef24c802fe864dfa9580914ce26df3035 (HEAD -> master, origin/master, origin/HEAD)
Author: Anthony Giretti <anthony.giretti@gmail.com>
Date:   Sat Jan 4 20:08:20 2020 -0500

    ajout fichier non d<C3><A9>sir<C3><A9>

commit b91039ff6a6c6a19fa6a6fef9c041135051afacd
Author: Anthony Giretti <anthony.giretti@gmail.com>
Date:   Sat Jan 4 19:47:32 2020 -0500

    Ajout fichier .gitignore

commit ad8afec317c34319e118b3eb39ad5731874897cd
Author: Anthony Giretti <anthony.giretti@gmail.com>
Date:   Sat Jan 4 19:39:12 2020 -0500

    remove gitignore
```

Important ! Il est nécessaire avec cette commande de saisir la date en format anglais.

## Chercher un commit et comparer les commits

Vous vous doutez bien que parfois il peut y avoir un problème dans une implication et nous voudrions investiguer où et quand une anomalie aurait pu être introduite. En se basant sur les commandes de vérification de l'historique, nous allons pouvoir comparer les commits entre eux et trouver l'anomalie.

Imaginons une anomalie dans un fichier dans le dernier commit et comparons ce fichier avec l'avant-dernier commit, il faudra d'abord récupérer l'ID des commits (l'ID court du commit suffit) avec la commande *git log –oneline -2* (l'option –option permet d'avoir moins d'information dans l'écran de sortie).

```
~\source\repos\GitDemoCommandLine [master ≡]> git log --oneline -2
e96aa11 (HEAD -> master, origin/master, origin/HEAD) ajout d'une anomalie
247e83e ajout fichier non d<C3><A9>sir<C3><A9>
~\source\repos\GitDemoCommandLine [master ≡]>
```

Maintenant que nous connaissons l'ID des commits à comparer lançons la comparaison entre les deux commit avec la commande *git diff e96aa11 247e83e*

```
~\source\repos\GitDemoCommandLine [master ≡]> git diff e96aa11 247e83e
diff --git a/GitDemoCommandLine/WeatherForecast.cs b/GitDemoCommandLine/WeatherForecast.cs
index 5354bf0..95cbfc8 100644
--- a/GitDemoCommandLine/WeatherForecast.cs
+++ b/GitDemoCommandLine/WeatherForecast.cs
@@ -10,6 +10,6 @@ namespace GitDemoCommandLine

         public int TemperatureF => 32 + (int)(TemperatureC / 0.5556);

-        public string Summar { get; set; }
+        public string Summary { get; set; }
     }
 }
~\source\repos\GitDemoCommandLine [master ≡]>
```

En rouge nous voyons ce qui a été remplacé et en vert ce qui a été introduit. Nous voyons donc l'anomalie qui cause l'échec de la compilation. Admettons désormais que vous êtes persuadés que c'est votre dernière modification ajoutée à l'index (mais sans en être sur) qui pose problème il est possible de comparer votre index à votre répertoire de travail avec la commande *git diff*.

Pour comparer votre répertoire de travail à votre dépôt local, vous pouvez le faire avec la commande *git diff HEAD*.

Enfin pour comparer votre index à votre dépôt local, utilisez la commande *git diff --cached*

## Revenir à un commit spécifique

Considérons maintenant que vous avez récupéré les commits d'autres collègues et qu'un des commits comporte des anomalies que vous avez probablement identifiées grâce à l'historique, et bien plutôt que de corriger une anomalie avec un autre commit, la bonne pratique voudrait qu'on annule le commit pour revenir à l'état précédent, sans l'erreur donc. En règle générale vous voulez tout d'abord vérifier le commit potentiellement coupable pour voir s'il produit les erreurs qui ont été constatées, vous pouvez tester ce commit avant de l'annuler. Pour le tester il vous suffit de vous positionner dessus en tapant la commande suivante : *git checkout e96aa11*. Votre répertoire de travail contient donc la supposée anomalie et vous la testez. Une fois certain de vôtre diagnostic vous voulez l'annuler, il vous suffit de faire *git revert unwantedcommitId* :

```
~\source\repos\GitDemoCommandLine [master ≡]> git revert e96aa11
```

Le message suivant va s'afficher :

```
Revert "ajout d'une anomalie"

This reverts commit e96aa11e014559069a866d7e786b7fd3620ae8b3.

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch master
# Your branch is up to date with 'origin/master'.
#
# Changes to be committed:
#       modified:   GitDemoCommandLine/WeatherForecast.cs
#
~
~
```

Faîtes **:q** pour sortir du fichier (vous pouvez laisser le commentaire tel quel) et ensuite faire un push :

```
~\source\repos\GitDemoCommandLine [master ↑1]> git push
Counting objects: 4, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 404 bytes | 404.00 KiB/s, done.
Total 4 (delta 3), reused 0 (delta 0)
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To https://github.com/AnthonyGiretti/GitDemoCommandLine.git
   e96aa11..0d5a4d4  master -> master
~\source\repos\GitDemoCommandLine [master ≡]>
```

Dernière vérification du fichier source qui posait problème avec la commande *git log problematicfile* :

```
~\source\repos\GitDemoCommandLine [master ≡]> git log GitDemoCommandLine/WeatherForecast.cs
commit 0d5a4d48bf4f5b79a5899abe7da7aa881a5c2fe1 (HEAD -> master, origin/master, origin/HEAD)
Author: Anthony Giretti <anthony.giretti@gmail.com>
Date:   Sat Jan 4 22:24:48 2020 -0500

    Revert "ajout d'une anomalie"

    This reverts commit e96aa11e014559069a866d7e786b7fd3620ae8b3.

commit e96aa11e014559069a866d7e786b7fd3620ae8b3
Author: Anthony Giretti <anthony.giretti@gmail.com>
Date:   Sat Jan 4 20:55:12 2020 -0500

    ajout d'une anomalie
```

## Conclusion

Nous venons de voir comment gérer l'historique du projet GIT, comparer les versions et annuler des commits si cela s'avère nécessaire.

Bien sûr il s'agit encore qu'un survol des possibilités qu'offre GIT, dans le prochain et dernier article consacré à GIT, nous verrons comment créer des branches de travail à partir d'une autre branche puis les supprimer, fusionner des changements provenant de diverses branches et gérer les conflits. Nous également comment gérer les tags. Enfin nous verrons comment rebaser une branche et voir l'utilité de la commande squash pour une meilleure lisibilité de l'historique de version.

**Olivier LOURME**
*Enseignant en Génie Électrique & Informatique à l'Université de Lille, doctorant au laboratoire CRIStAL UMR 9189*
**Twitter :** @OlivierLourme
**Medium :** medium.com/@o.lourme

# ESP32, Mongoose OS et GCP Cloud IoT Core : un trio efficace et sûr pour l'IoT

Partie 1

*Dans un contexte de développement autour des objets connectés, nous présentons ici une solution « Objet / OS / plateforme cloud » efficace en terme de temps de développement tout en étant « secure by design ». L'objet est un* **ESP32** *du chinois Espressif, son OS (et outillage de provisionnement) est* **Mongoose OS** *et la plateforme retenue est ici* **Google Cloud Platform** *(GCP) avec sa brique « Cloud IoT Core ».* **Firebase** *sera également utilisé.*

## CONTEXTE ET CAS D'USAGE ÉTUDIÉ

L'acronyme **IoT** (*Internet of Things*, ou Internet des Objets) fait le buzz depuis plusieurs années maintenant. Et si on faisait une petite piqure de rappel ?

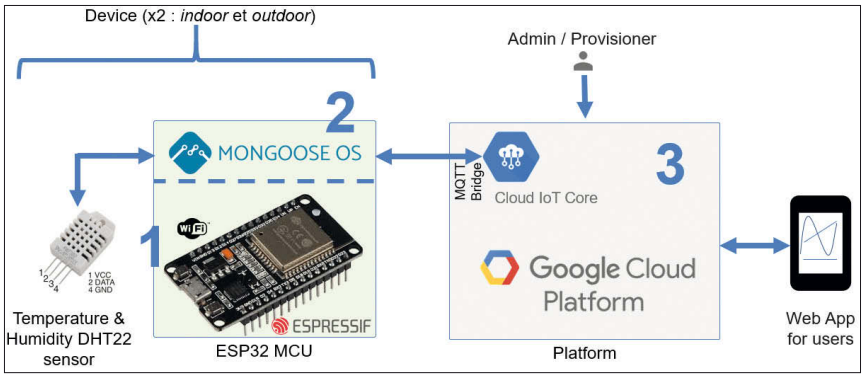
Un **objet** (*thing*, *device*) est un microcontrôleur doté d'une capacité de **communication**, souvent sans fil (technologies Wi-Fi, Bluetooth, LoRa, etc.), auquel sont associés des **capteurs** et des **actionneurs**. Ainsi, l'objet est souvent un nœud dans un **réseau** et il est capable d'envoyer à d'autres nœuds (passerelle, pont, *broker* ou parfois un autre objet) les mesures qu'il réalise et de piloter ses actionneurs en fonction des consignes qu'il reçoit d'un autre nœud. Les topologies de réseau mettant en œuvre ces objets sont variables selon la technologie de communication utilisée. Très souvent, les données issues ou à destination de ces objets se retrouvent à emprunter Internet, d'où l'expression « Internet des Objets ». Un des challenges de l'IoT vient du fait que les objets sont des systèmes à ressources fortement contraintes mais qu'il leur faut quand même respecter les règles de sécurité liées à Internet, très consommatrices de ressources.

Il semblerait que l'agitation autour de l'IoT soit justifiée : selon l'Usine Digitale (*1*), les prévisions 2020 font état de 10 milliards d'objets connectés « purs » (hors ordinateurs, tablettes et smartphones) avec une croissance de ce nombre de 15% par an pour les prochaines années. En termes de marché mondial, celui-ci est déjà actuellement de plusieurs centaines de milliards de dollars. 75% de ces objets se connectent par des technologies de réseaux personnels (Bluetooth, ZigBee) ou de réseaux locaux (Ethernet, Wi-Fi), même si les technologies LPWAN (LoRa, SigFox, NB-IoT) sont promises à se développer fortement.

Quand on déploie un parc de plusieurs dizaines d'objets, trois préoccupations majeures doivent nous animer :

- **Le provisionnement des objets** (*id est* le process allant de l'achat à celui des conditions opérationnelles, en passant par le *flash* du *firmware* dans la mémoire, la génération de clés asymétriques pour l'authentification, etc.) doit être mené avec un souci d'automatisation. Ainsi, outre **un outillage (*tooling*) plutôt haut niveau** pour ce provisionnement, il faut **une administration centralisée** et scalable pour ces objets c'est-à-dire typiquement **une plateforme *cloud* pour l'IoT**.

(*1*) http://bit.ly/32eeXCB




1 *L'objet, son OS et la plateforme*

- **La sécurité** doit être au cœur de tous les éléments de la chaine : un objet ne doit pouvoir livrer des informations quand il est physiquement attaqué (mots de passe Wi-Fi pour prendre un exemple simple) ; de même des mécanismes d'authentification sont à prévoir et les échanges objets-plateforme doivent être confidentiels. Il faut être ***secure by design***, après c'est trop tard !
- La plateforme *cloud* retenue doit **proposer des outils innovants** permettant d'exploiter intelligemment les données fournies par les objets (tableaux de bord, traitement *big data*, intelligence artificielle, notifications, etc.). En outre, elle doit **imposer un cadre** pour le développement et la sécurité, évitant bien des interrogations.

Pour illustrer le respect de ces trois aspects par notre solution, nous nous fixons le **cas d'usage** classique du **monitoring de la température et de l'humidité en différents points d'un bâtiment**. Les données doivent être persistées dans une base de données et tracées en temps réel. En chacun des points de mesure se trouve un objet disposant d'une connectivité Wi-Fi. Nous ne mettrons en œuvre que deux objets, nommés *indoor* et *outdoor* mais nous aurons toujours à cœur d'obtenir un déploiement facile, qu'il y ait deux capteurs comme ici, ou beaucoup plus.

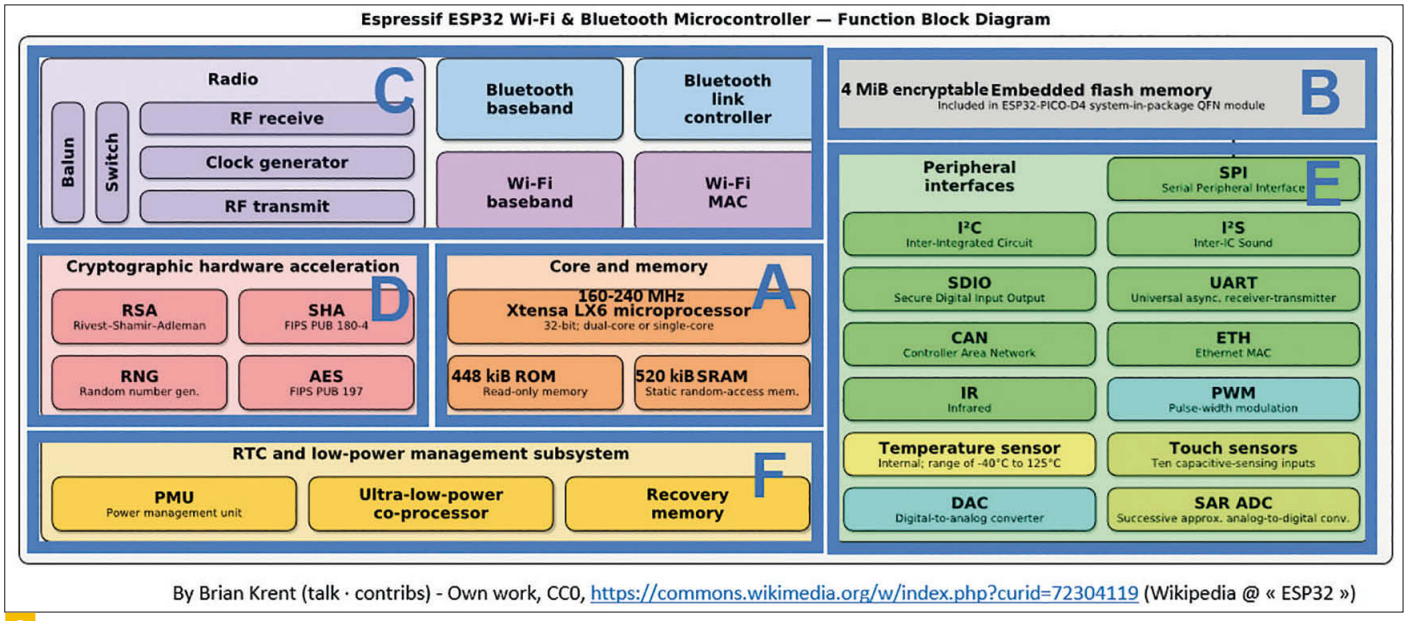
Le contexte de la réalisation peut-être :

- un projet d'étudiants à la fin d'un module d'initiation à l'IoT,
- un *Proof Of Concept* ou un prototypage rapide par des ingénieurs de bureau d'études,
- une base sérieuse pour un cas de production.

## LA TRILOGIE RETENUE

Sur la **Figure 1**, on distingue les trois éléments (1, 2 et 3) de notre solution :

- **Un objet disposant d'une connectivité Wi-Fi** pour se connecter




By Brian Krent (talk · contribs) - Own work, CC0, https://commons.wikimedia.org/w/index.php?curid=72304119 (Wikipedia @ « ESP32 »)

2 *Décomposition fonctionnelle de l'ESP32*

à la plateforme et **de modules périphériques** pour s'interfacer avec ses capteurs. Nous avons retenu l'**ESP32** (2) d'Espressif, successeur de l'ESP8266. Une recherche ESP32 sur en.wikipedia.org donne des cas d'utilisation en production de cette puce. En continuant de chercher sur le web, on voit aussi que les principales plateformes *cloud* (Google Cloud Platform, Amazon Web Services (AWS), Microsoft Azure, IBM Watson) l'utilisent dans certaines de leur démonstration IoT. Sur la **Figure 1**, l'ESP32 correspond juste à grosse puce argentée. Il est ici monté en **DEVKIT v1**, offrant un convertisseur USB-UART, un régulateur de tension 5V vers 3.3 V, deux LEDs, deux boutons et deux rangées de pattes au pas standard. On peut trouver ce DEVKIT sur les sites chinois à moins de 4 €. Le DHT22 est quant à lui un capteur de température et d'humidité bon marché (2 €), très diffusé. Il est de type *one wire*, c'est-à-dire qu'outre les alimentations, une seule patte est nécessaire ; celle-ci est bidirectionnelle et véhicule tantôt de la commande, tantôt de la donnée. Pour ne pas perdre son temps, il faut bien sûr utiliser une bibliothèque pour le piloter. Pour des capteurs plus précis (et plus chers), on peut éventuellement s'intéresser aux composants de la série SHT3X ou au BME280.

- **Un « OS »**, se situant au-dessus de l'objet afin que ce dernier dialogue *via* des API de haut niveau avec ses capteurs ainsi qu'avec la plateforme *cloud*, de façon sécurisée. Cet OS doit également venir accompagné d'un outillage pour provisionner facilement l'objet. Comme nous le verrons, dans ce double contexte, **Mongoose OS** tire bien son épingle du jeu et amène **des gains conséquents en temps de développement**. Il est adapté à l'enseignement, au prototypage rapide et à la production. Il est bien-sûr cautionné par les principales plateformes *cloud* (3) et on trouve sur le *net* des exemples d'utilisation en production (4).
- **Une plateforme *cloud*** pour administrer les objets et exploiter les mesures qu'ils fournissent. Nous utiliserons **Google Cloud Platform** (GCP) et sa brique **Cloud IoT Core** (5). À noter que le sens inverse de transfert d'information est aussi possible, la plateforme pouvant envoyer une configuration ou une commande à un objet.

# L'OBJET ET SON OS

## Le microcontrôleur ESP32

De technologie 40 nm, l'ESP32 est un microcontrôleur 32 bits Wi-Fi / Bluetooth performant et richement doté. La **Figure 2** donne un schéma des différents blocs fonctionnels. Nous ne commentons ici que ce qui est nécessaire à notre cas d'usage.

(A) L'ESP32 est à **double cœur** (un des deux peut être consacré à la communication) et est **cadencé à 160 MHz**. La ROM sert pour le *boot* et les *core functions* (celles du Wi-Fi par exemple). Il dispose d'environ 0.5 Mo de RAM.

(B) L'ESP32 utilisé dispose d'une **mémoire flash SPI de 4 Mo, chiffrable**. Cette mémoire héberge le programme et/ou un OS, un éventuel site web, etc. Nous verrons dans la partie sur la sécurité en quoi le fait qu'elle soit chiffrable est intéressant.

(C) Ce bloc concerne la **partie radio** et gère Wi-Fi et Bluetooth (4 et BLE).

(D) Un bloc d'**accélération matérielle pour le chiffrement** est disponible, ce qui sera très utile pour la sécurité des échanges avec la plateforme, l'authentification, etc.

(E) On trouve aussi tous les **modules périphériques** traditionnels (PWM, ADC, etc., en grand nombre), mais ici seule une des trois UARTs (pour communiquer avec le PC de développement) et une GPIO (pour la connexion avec le capteur DHT22) seront utilisées.

(F) Enfin, on voit qu'un bloc ***low-power management*** est présent. Nous ne l'avons pas mis en œuvre pour une raison que l'on expliquera plus loin. Il est évident qu'il faudrait s'y pencher si on est animé par des considérations d'autonomie énergétique. On pourrait par exemple mettre l'objet en veille entre deux mesures. Pour l'instant ce n'est pas notre cas et nous avons une connexion Wi-Fi permanente entre l'objet et son routeur. Cela participe à une

(2) https://www.espressif.com/en/products/hardware/esp32/overview

(3) https://mongoose-os.com/about.html

(4) https://mongoose-os.com/case-studies/shelly.html

(5) https://cloud.google.com/iot-core/

consommation moyenne mesurée de 90 mA sous 5 V, ce qui est loin d'être négligeable. Avec une batterie (5 V - 2000 mAh), on dispose d'une autonomie de : 2000 / 90 = 22 heures, soit moins d'une journée ! Il faudra de ce fait alimenter nos objets par des chargeurs de téléphone.

## Mongoose OS

### ESP-IDF, freeRTOS

Quel SDK (*Software Development Kit*) accompagne officiellement l'ESP32 ? Espressif propose son *IoT Development Framework* nommé **ESP-IDF** ([6]). Il s'appuie sur **freeRTOS**, OS temps réel bien connu. Cela dit, ces environnements, bien que très performants et complets pour les spécialistes, sont longs et complexes à prendre en main pour des étudiants ou pour des ingénieurs souhaitant valider une piste. En effet, une très bonne connaissance du C et des API disponibles ainsi que des bases sérieuses en chaine de développement sont nécessaires. Globalement :

- Provisionner un objet suppose souvent la génération de clés et l'hébergement de celles-ci aux bons endroits. De même, une mise à jour du programme suppose un nouveau ***build*** suivi d'un ***flash*** du *firmware* nouvellement généré. Bref, ces étapes classiques représentent beaucoup d'actions manuelles et de temps passé, même si du *scripting* peut améliorer les choses.
- Les APIs sont de bas niveau et il y a donc beaucoup de code à écrire. Il suffit de voir par exemple la documentation de freeRTOS en ce qui concerne la publication de messages par le protocole MQTT (*7*).

**Remarque :** dans notre cas d'usage, nous verrons qu'un objet publie ses mesures vers le point d'entrée de la plateforme grâce au protocole MQTT. Si vous n'êtes pas au fait de ce protocole, c'est le moment de vous y mettre car c'est sans doute LE protocole de l'IoT. Une brève vidéo du DEVOXX FR 2017 en fait un tour d'horizon : (*8*).

### Mongoose OS

**Mongoose OS** repose sur l'idée que dans l'IoT, beaucoup d'étapes se répètent d'un projet à l'autre tant au niveau provisionnement que programmation. Il se propose donc de les exposer avec un haut niveau d'abstraction ou de les effectuer automatiquement. Il se définit donc comme un *framework* IoT.

Il propose **des outils de haut niveau pour le provisionnement**, qu'il appelle des commandes **mos** (*upload* d'un fichier vers l'objet, génération et dépôts de clés, *reboot*, etc.).

Au niveau **programmation**, Mongoose OS propose deux modèles :

- Une programmation en C avec des APIs plus abstraites que celle de freeRTOS. Ce n'est pas cette fonctionnalité de Mongoose OS que nous avons explorée, car la syntaxe reste ardue pour ceux qui n'ont jamais fait de C.
- **Une programmation en JavaScript (JS) avec des APIs de haut niveau**. En fait, le développeur écrit ses fichiers **js** (le principal se nommant **init.js**) et les *uploadent* vers l'objet à l'aide de la commande **mos** appropriée. Un *reboot* plus loin et **le JS est interprété par le moteur JS de nom mJS** (*9*) tournant sur l'objet. Pour reprendre le même exemple qu'avec freeRTOS, une publication MQTT en JS avec Mongoose OS vaut le coup d'œil eu égard à son extrême compacité (*10*).

C'est évidemment ce dernier type de programmation que nous allons retenir. Les programmes deviennent bien plus concis ! Il va de soi que mJS n'est qu'un petit sous-ensemble de JS mais des méthodes comme respectivement **JSON.stringify** ou **JSON.parse** sont disponibles, ce qui est très utile quand l'objet veut respectivement publier en JSON sur un *topic* MQTT ou quand il veut analyser un message JSON d'un *topic* auquel il est abonné.

### Bilan des différents OS

Dans le tableau **Tableau 1**, on dresse une petite synthèse freeRTOS / Mongoose OS même si l'on est bien conscient qu'il ne s'agit pas de comparer ces deux outils.

**Remarque :** pour revenir à l'exploitation du bloc fonctionnel ***low-power management*** de l'ESP32, Mongoose OS n'a pas d'API pour le gérer. Il est possible cela dit d'intégrer, au sein d'un développement Mongoose OS, cette gestion avec ESP-IDF. Toutefois, le projet cesserait d'être *cross-device*.

Pour conclure sur Mongoose OS, il est évident que celui-ci ne va pas remplacer freeRTOS (surtout qu'AWS a acquis ce dernier en 2017…) mais il a su proposer des outils et des APIs haut niveau dédiés à l'IoT que freeRTOS n'offrait pas. Rien que pour cela, cela vaut la peine de s'y pencher. Un dernier point : il a une communauté active et une bonne documentation.

**La suite dans le numéro 238.**

(*6*) https://docs.espressif.com/projects/esp-idf/en/stable/get-started/

(*7*) https://www.freertos.org/mqtt/preconfiguredexamples.html

(*8*) https://youtu.be/C9dKYqTfuqE

(*9*) https://github.com/cesanta/mjs

(*10*) http://bit.ly/2NjMdEz

**Tableau 1. freeRTOS et Mongoose OS**

| | freeRTOS | Mongoose OS |
|---|---|---|
| **création** | 2003, 40 architectures supportées | 2013, 7 microcontrôleurs supportés (ESP, ST, TI), *cross-device* |
| **licence** | Open source, licence MIT | Open source, licence Apache 2.0 ou licence commerciale (pour support + possibilité de mise à jour *Over The Air*) |
| **langage** | C avec apprentissage longAPIs de bas niveau, programmes longs | C ou…… mJS interprété : apprentissage courtAPIs de haut-niveau, programmes courts, prototypage rapide |
| **déploiement et mise à jour du *firmware*** | long : avec ESP-IDF, *build* et *flash* | rapide : par 2 commandes **mos** : *upload* de **init.js** + *reboot* |
| **provisionnement d'un objet(en plus du *firmware*)** | outillage inexistant : création/gestion de clés, enregistrement auprès de GCP scripts à écrire | outillage de haut-niveau :une seule commande **mos** pour création/gestion de clés et enregistrement auprès de GCP |
| **authentification par *JSON Web Token* d'un objet auprès de GCP** | code à écrire | automatique |
| ***plug-in* Visual Studio Code** | oui, | oui, développé par l'éditeur (*a*)<br>(*a*) https://mongoose-os.com/docs/mongoose-os/quickstart/ide.md |

**Julien Chomarat**
*Senior Software Engineer chez Microsoft. Développeur dans l'âme, j'aide les entreprises du secteur bancaire à s'approprier les services Azure à travers des projets innovants, en mode co-développement.*
https://github.com/jchomarat

# Développer sous Linux depuis une machine Windows 10 !

***Entre 2016 et aujourd'hui, la proportion de machines virtuelles Linux provisionnées dans Microsoft Azure est passée de 25 % à un peu plus de 50 %. Linux est donc passé devant Windows !***

Il existe par conséquent un réel besoin de pouvoir faire du développement sur un environnement Linux, et ce, pour diverses raisons :

- Flexibilité d'installation d'environnement de développement via gestion de paquets
- Facilité de scripts d'automatisation des machines (installation / environnement)
- Accès via Terminal à une multitude de commandes et d'outils

Cette liste n'est pas exhaustive. Le constat actuel est que Windows n'est pas suffisamment flexible pour mettre en place tout ce que venons de voir. Alors oui, nous avons bien entendu l'approche « virtualisation » - via Hyper-v ou VirtualBox (pour ne citer qu'eux) permettant d'installer la distribution de votre choix. Nous avons également l'approche conteneurs, grâce à Docker par exemple. Cependant, cela nécessite plus de ressources matérielles, de l'administration et de la maintenance. La plupart des développeurs veulent s'affranchir de tout cela, pour juste avoir « à coder ». Nous allons aborder dans cet article une nouvelle approche, plus simple à mettre en place, et permettant de provisionner un vrai environnement Linux de développement, depuis un système Windows 10.

## Quelle alternative à une machine virtuelle ?

En automne 2017, Microsoft a introduit dans Windows 10 (*build* 16215) une fonctionnalité, appelée **W**indow **S**ubsystem for **L**inux, ou WSL. Ce premier jet, en version 1.0, a été un vrai succès auprès de la communauté de développeurs. Ce succès fut tel que Microsoft a décidé d'investir dessus, et vient de sortir la version 2.0 (été 2019), avec une architecture sous le capot complètement refondue, et bien plus optimisée.

*WSL* permet d'installer des distributions Linux complètes, depuis le store. À ce jour, les distributions suivantes sont disponibles : Ubuntu, OpenSuse, Suse, Kali, Debian et Alpine en version gratuite, ainsi que Fedora et Pengwin en version payante. Différentes versions de ces distributions sont disponibles (1).

Une fois notre distribution Linux installée, nous allons voir quelques outils indispensables à tout développeur pour profiter au maximum de notre « Linux embarqué ». Nous développerons enfin une API web simple – l'idée étant de voir comment développer sous « Linux ».

## Une distribution Linux dans WSL, sans virtualisation ?

Une distribution Linux installée sur Windows 10 n'est pas une machine virtuelle au sens où nous l'entendons, avec toutes les contraintes que cela apporte (long à démarrer, consommateur de ressources, nécessitant de l'administration, etc.). WSL reste une virtualisation, mais dans les couches basses du système d'exploitation. Depuis l'été 2019 et la version 2.0 de WSL, Windows 10 inclut un noyau Linux en parallèle au noyau Windows. Microsoft est partie d'une version stable du noyau Linux (4.19 à ce jour), et a apporté des modifications pour le faire « entrer » dans Windows. Si vous aviez commencé à manipuler *WSL 1.0*, je vous invite fortement à migrer vos distributions en *WSL 2.0* (ce qui n'est pas fait par défaut lors de la mise à jour de Windows). Je vous invite d'ailleurs à vous reporter à l'article paru dans le Programmez d'octobre 2019 (2) pour plus de détails sur cette architecture *WSL* et la façon dont cela fonctionne au sein de Windows.

Mais alors, que cela m'apporte-t-il ? Hormis pour du développement, que nous verrons dans un instant, vous aurez une distribution Linux complète. Attention, cela reste une version Linux en « ligne de commandes ». Vous n'aurez pas d'interface graphique type Gnome/KDE ou autre. Cela reste une version Linux en mode console. Des initiatives fleurissent sur la toile pour faire tourner des applications Linux graphiques sous Windows – mais l'idée derrière WSL n'est pas, du moins à ce jour, le but.

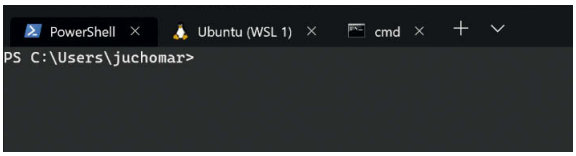
Grâce à *WSL*, nous pourrons avoir accès aux commandes de base dont tout développeur à besoin : ssh, grep, vi, git etc. Depuis ce système Linux, vous aurez également accès aux disques Windows (montés par défaut dans /mnt) et vice et versa. Vous pourrez même éditer vos fichiers « Linux » depuis l'explorateur Windows – ce qui, ceci dit en passant, était fortement déconseillé en *WSL 1.0* – mais possible maintenant ! Ouvrez votre console Linux, et tapez la commande « explorer.exe. » Eh oui, cela veut dire que je peux exécuter des programmes « Windows » depuis mon instance Linux ? Pas tout à fait, vous pouvez juste lancer des programmes Windows, qui s'exécuteront bien dans le contexte Windows, mais depuis votre instance Linux.

Et enfin, dernière chose, et c'est là aussi que la magie opère : vous pouvez accéder en réseau (protocole http par ex.) à votre instance Linux, en utilisant « localhost ». Un pont réseau interne entre ces deux systèmes est créé au lancement de l'instance Linux. Cela est très intéressant dans le cadre de développement web, dans son sens le plus large.

## De quoi d'autre ai-je besoin ?

Vous avez activé *WSL* sur votre PC et installé une distribution, Ubuntu par exemple. Pour démarrer Ubuntu, il suffit de lancer l'application Ubuntu (depuis le menu *Démarrer*), qui en fait, lance une invite de commande Ubuntu en exécutant la commande « wsl.exe -d NomDeLaDistribution ». C'est comme si vous vous connectiez en *ssh* sur un serveur Linux distant, mais là, vous ne sortez pas de votre machine.

Et là, les choses deviennent plus compliquées : vous vous retrouvez avec un terminal pour Ubuntu (ou toute autre distribution installée), un terminal, le fameux shell « cmd » pour Windows et un terminal PowerShell ! Ça fait beaucoup de terminaux tout ça !
Le premier outil à installer sur votre machine Windows est tout simplement le nouveau *Windows Terminal* (3)… Sous ce nom « simple », se cache un formidable outil *open source* permettant d'agréger en une fenêtre tous les terminaux. Attention, ce projet est encore en version *preview*, mais, au moment de l'écriture de cet article, la version 0.5 est parfaitement stable et exploitable au quotidien.

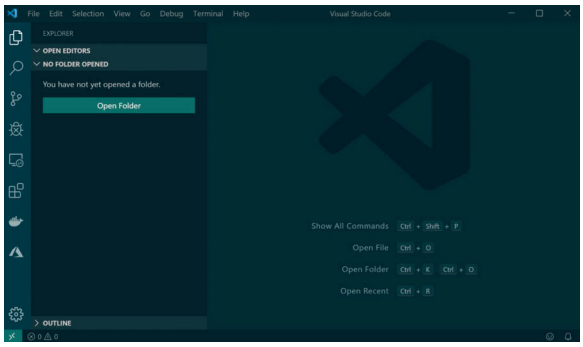



*Le nouveau Windows Terminal*

Comme vous constatez ci-dessus, chaque onglet peut être un shell différent : *cmd*, *PowerShell*, *WSL 1.0* ou *WSL 2.0*. Vous pouvez même avoir le *Shell Azure* directement, sans passer par votre navigateur (4).

### Dois-je développer avec vi ou nano ?

Rien ne vous l'interdit, avec une distribution Linux à portée de main… mais, nous apprécions tout de même notre confort !
Si vous ne l'avez pas déjà, je vous invite à installer *Visual Studio Code* (5). Ai-je besoin de le présenter : *Visual Studio Code* (*vscode* pour les intimes) est un éditeur/IDE, également open source, développé par *Microsoft*, maintenu et étendu via un système de paquets par toute une communauté de passionnés.




*Visual Studio Code*

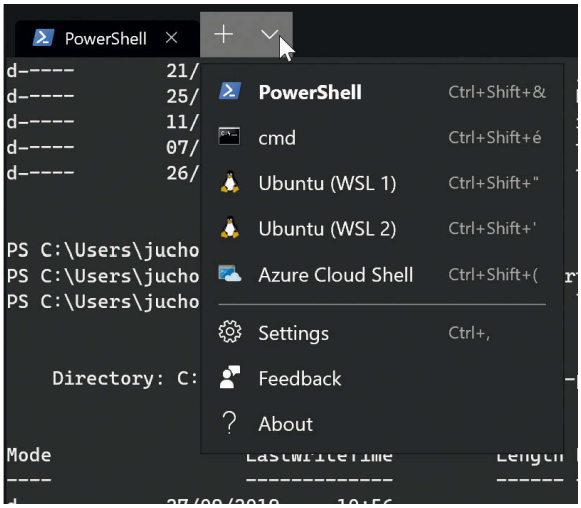
Installez au passage les plug-ins suivants : *Remote WSL* (6) (toujours en *preview* au moment de l'écriture de cet article, mais tout comme le *Windows Terminal*, très stable et exploitable) ainsi que le plug-in *Python* (que nous utiliserons dans notre exemple plus bas).

## Je veux coder maintenant !

Nous avons une distribution Linux qui tourne, un puissant terminal et un éditeur riche. Pour utiliser tous ces outils, je vous propose de développer un web service simple en Python. Le fonctionnel de ce service n'est pas important, car ce n'est pas ça que je cherche à montrer – mais pour le *fun fact*, il sera autour de *Star Wars*. Nous allons donc tirer parti de tout ce que nous avons installé – et ainsi faire du « développement » dans un environnement Linux depuis un environnement *Windows 10*.
Je vais partir du postulat que vous avez installé *WSL* (1.0 ou 2.0, peu importe pour la suite, comme je vous disais plus haut, ce sont essentiellement les performances qui ont été améliorées). Pour un souci de simplification textuel, je vais également supposer que la distribution installée dans *WSL* est *Ubuntu* (mais bien entendu que la suite fonctionnerait parfaitement sur une autre distribution, il faudra juste adapter les commandes de gestion des paquets au niveau du système).

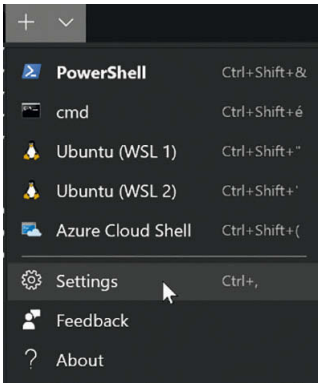
### Étape 1 : Environnement de développement.

Lançons notre Windows Terminal, et assurons-nous de pouvoir accéder à *Ubuntu*. Pour cela, cliquez sur le menu déroulant du terminal pour voir si WSL est disponible




*Lancement du shell Ubuntu*

Si ce n'est pas le cas, nous allons l'ajouter dans la configuration de Windows Terminal.
Sélectionnez « Settings » - qui va ouvrir un fichier dans votre éditeur *json* par défaut, *vscode* dans mon cas.




*Paramètres de Windows Terminal*

Le fichier de configuration est « brut » en *json* (mais l'équipe produit de Windows Terminal a déjà annoncé un éditeur graphique pour une version future). Pour ajouter le *shell Ubuntu*, ajouter le nœud suivant dans le tableau « profiles ».

[code01 - vscode config file.txt]

```
{
        "acrylicOpacity" : 0.5,
        "background" : "#2E2E2E",
        "backgroundImage" : "C:/…/ms_loves_linux.png",
        "backgroundImageOpacity" : 0.89999997615814209,
        "closeOnExit" : true,
        "colorScheme" : "Campbell",
        "commandline" : "wsl.exe -d Ubuntu",
        "cursorColor" : "#FFFFFF",
        "cursorHeight" : 25,
        "cursorShape" : "vintage",
        "fontFace" : "Cascadia Code",
        "fontSize" : 9,
        "guid" : "{2c4de342-38b7-51cf-b940-2309a097f518}",
        "historySize" : 9001,
      "icon" : "ms-appx:///ProfileIcons/{9acb9455-ca41-5af7-950f-6bca1bc9722f}.png",
        "name" : "Ubuntu (WSL 2)",
        "padding" : "10, 10, 10, 10",
        "snapOnInput" : true,
        "startingDirectory" : "%USERPROFILE%",
        "tabTitle" : "Ubuntu (WSL 2)",
        "useAcrylic" : false
    }
```

Les points importants à souligner sont les suivants :

- backgroundImage : chemin d'accès vers l'image de fond que vous souhaitez dans votre terminal
- commandLine : la commande à lancer pour ce profile – ici « wsl.exe -d Ubuntu ». Elle permet de lancer la distribution nommée Ubuntu dans WSL
- name : nom du profile dans le menu contextuel
- tabTitle : nom apparaissant dans l'onglet, lorsque ouvert dans le terminal
- fontFace : la font à utiliser. Petite remarque, Cascadia Code (7) est une nouvelle font – livrée par défaut dans Windows Terminal (dès la version 0.5). Elle est adaptée pour du code, avec notamment des « ligatures » (par ex. = puis > se transforme en flèche)

Assurons-nous maintenant que Python3.6 est installé sur notre instance Ubuntu.

```
/> sudo apt-get install python3.6
```

## Étape 2 : création de notre projet

Depuis le *shell Ubuntu*, naviguons jusqu'au « home » de l'utilisateur, et créons un nouveau dossier « starwars »

```
/> cd
/> mkdir starwars
/> cd starwars
```

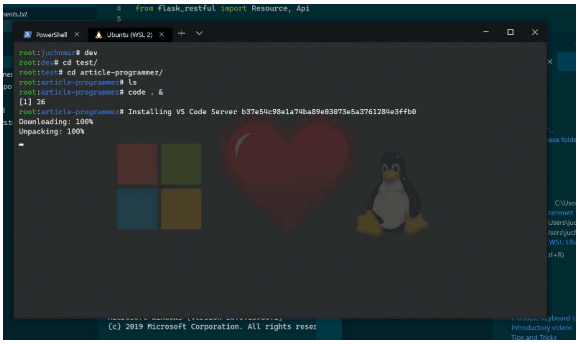
Nous allons maintenant créer un environnement virtuel Python pour installer les dépendances « localement » :

```
/> python3 -m venv env
 /> source env/bin/activate
Lançons vscode en utilisant la commande ci-dessous
/> code. &
```

La commande « code. » permet d'ouvrir *vscode* pour le dossier courant (donc */home/$USER/starwars*).

Deux choses intéressantes se produisent à ce niveau :

- Il y a un réel pont entre les deux systèmes fichiers, car vscode tourne sous Windows, tout en ouvrant un répertoire sous Linux
- Etant donné que *vscode* détecte l'ouverture d'un dossier depuis *WSL* – et comme nous avons installé précédemment le plug-in *Remote WSL* - la version serveur s'installe. Ce que nous avons installé précédemment était la version « cliente », ce dernier se charge tout seul d'installer son pendant serveur.




*Installation du Remote plugin, côté serveur*

Nos deux systèmes sont maintenant liés.

## Étape 3 : code

Comme mentionné plus haut, le fonctionnel de notre service importe peu – l'idée est de « coder ». Nous allons donc créer une web API en *Python*, en utilisant la librairie « *Flask* ».

Créons un premier fichier depuis *vscode* nommé « requirements.txt » et ajoutons la ligne suivante :

```
Flask==0.12
```

Nous indiquons à notre programme Python que nous aurons besoin de la dépendance *Flask*.

Créons un deuxième fichier, appelé *starwars.py*. Ce fichier contiendra le code de l'api. Nous allons ici créer un *endpoint* appelé « /characters » qui renverra la liste des personnages (des premiers films).

Le code (très simple) complet se trouve ci-dessous :

[code02 - python webapi.txt]

```
#!/usr/bin/python3
from flask import Flask, jsonify

app = Flask(__name__)

@app.route("/characters", methods=["GET"])
def caracters():
  return jsonify({
        "Characters":[
          "Luke",
          "Leia",
          "Han",
          "Chewbacca"
        ]
    })
```

```
if __name__ == '__main__':
  app.run(port=1235, debug = True)
```

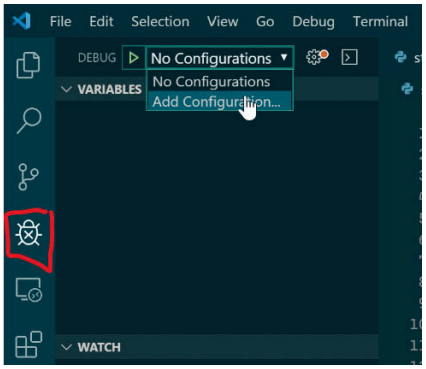
La méthode *characters* renvoi un tableau *json*, et s'accède en protocole *GET*. La dernière ligne va démarrer un serveur web et écouter sur le port *1235*.
Installons la dépendance *Flask* :

```
/> pip install -r requirements.txt
Et enfin, lançons notre serveur web :
/> python3 starwars.py
```

Nous avons un serveur web qui démarre sur le port 1235. Ce serveur tourne sur mon instance Linux, vais-je donc pouvoir y accéder depuis mon navigateur, ou un outil de requêtes API, tel que Postman ? La réponse est oui, bien entendu ! Un réseau existe entre les deux systèmes, et un *mapping* est fait sur ce même port.
Lançons donc notre navigateur préféré et tapons l'url suivante : http://localhost:1235/characters (l'appel étant un GET, il sera bien exécuté par le navigateur – et cela marchera également depuis Postman ou l'excellent plug-in vscode Rest Client (8)). Petite remarque au passage : toutes les commandes précédentes sont effectuées depuis Windows Terminal, dans l'onglet *Ubuntu* – mais, si vous ouvrez le terminal inclus dans *vscode*, vous vous retrouvez bien dans le bon répertoire, dans la couche Linux exécutée dans *WSL*, et surtout, dans l'environnement virtuel Python. Vous avez donc le choix. Personnellement, je préfère garder ma fenêtre de code en grand et basculer sur le *Windows Terminal* quand j'en ai besoin.

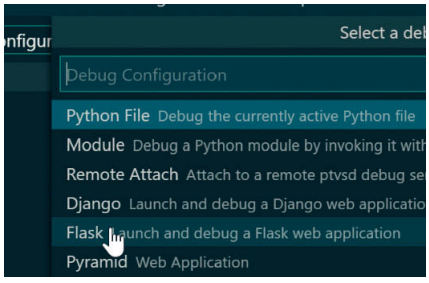
### Étape 4 : débug pas à pas

Outre le débug « à l'ancienne » à coup de « print() » qui affiche les informations demandées dans la console, nous avons, grâce au plug-in *vscode Remote WSL*, la possibilité de faire du débug pas à pas de notre programme. C'est également une des forces de ce plug-in. Basculons dans la section « debug » de *vscode* (ctrl+shift+d)




*Vscode ajout d'une configuration pour le débug.*

Et ajoutons la configuration Flask, en précisant comme point d'entrée notre fichier starwars.js



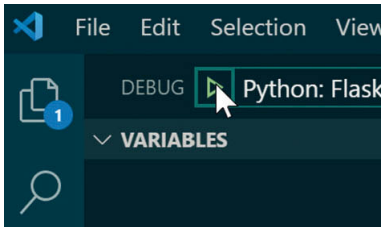

*Vscode, définition de Flask comme configuration de débug.*

Ajoutons la propriété « *FLASK_RUN_PORT* » pour rester sur le même port 1235

```
"configurations": [
    {
        "name": "Python: Flask",
        "type": "python",
        "request": "launch",
        "module": "flask",
        "env": {
            "FLASK_APP": "starwars.py",
            "FLASK_ENV": "development",
            "FLASK_DEBUG": "0",
            "FLASK_RUN_PORT": 1235
        },
        "args": [
            "run",
            "--no-debugger",
            "--no-reload"
        ],
        "jinja": true
```

*Vscode ajout du port d'écoute pour le débug.*

Tout est prêt. Posons un point d'arrêt dans la méthode *characters*, sur le « *return* » par exemple, et lançons-le debugger en cliquant sur le bouton « start debugging », ou F5 sur votre clavier.




*Vscode lancement du débuggeur*

Nous pouvons à présent faire du pas à pas dans *vscode*, depuis une application *Python* qui tourne sous Linux. Vous pouvez d'ailleurs inspecter, depuis la fenêtre de débug, toutes les propriétés disponibles.

## Conclusion

Nous avons vu qu'il est maintenant facile de faire du développement « Linux » depuis Windows 10. Nous retrouvons tous les outils attendus : système Linux complet, terminal et éditeur de code. Nous sommes partis sur un exemple Python, mais cela reste la même approche pour faire du Node.js, du DotNetCore ou tout autre langage tournant sous Linux. L'avantage de cette approche est que nous n'avons pas à installer sur notre machine Windows tous les « runtimes » de développement, de modifier les « *paths* », etc. Nous pouvons également avoir plusieurs instances Linux installées sous *WSL*, une par « environnement » de développement – et c'est typiquement ce que j'ai sur ma machine. *Windows Terminal* est encore en version bêta – mais il ne cesse de proposer de nouvelles fonctionnalités. Il en est de même pour *WSL* – la version 2.0 est toute fraîche, mais elle va encore évoluer et s'améliorer.

### Ressources

(1) https://docs.microsoft.com/fr-FR/windows/wsl/install-win10
(2) https://www.programmez.com/magazine/programmez-233-pdf
(3) https://github.com/microsoft/terminal
(4) https://devblogs.microsoft.com/commandline/the-azure-cloud-shell-connector-in-windows-terminal/
(5) https://code.visualstudio.com/
(6) https://marketplace.visualstudio.com/items?itemName=ms-vscode-remote.remote-wsl
(7) https://github.com/microsoft/cascadia-code
(8) https://marketplace.visualstudio.com/items?itemName=humao.rest-client

**Pauline Iogna**
*Développeur sénior*
**Lunatech France**
*Duchess France Leader*

# Découplez vos applications avec l'architecture hexagonale

Partie 2

*Qui n'a jamais rencontré des problèmes lors de montées de version, de migrations de base de données, ou encore de changements de framework ou de libary ? Qui ne s'est jamais retrouvé coincé dans un refactoring tant la complexité du code est importante ? Ou dans la difficulté de mettre à jour une règle de gestion ?*

**Suite de notre dossier.**

## Le découplage se fait par l'utilisation d'interfaces (les ports) et par la séparation du modèle

Le modèle unique entre la partie métier et le reste de l'application crée un couplage fort. Avec des frameworks comme Play ou Spring, dans une approche où les entités sont riches. Cela simplifie le code mais crée un couplage fort entre le code métier, et le code technique par les annotations ou des classes héritées. 9

Par exemple des annotations permettant de faire de la validation dans les frameworks MVC (Modèle Vue Controller).

En regardant les imports, on observe que la classe ci-dessous dépend du framework web playframework.

Coté base de données, des annotations peuvent être utilisées pour mapper les entities aux tables et aux colonnes des tables. 10

Ici les imports indiquent que le framework utilisé est *Ebean* pour mapper la base de donnée. Il n'y a de dépendances que vers des classes permettant de traiter la persistance ou le mapping vers la base. Si votre choix est de construire une application sur les principes de l'architecture hexagonale, alors votre métier doit contenir essentiellement des *Value Objects*. C'est à dire, des classes simples dont l'égalité n'est pas basée sur un identifiant mais sur la valeur des champs de l'objet. Les values objects étant définis par leur état, doivent être immutables. On observe dans l'exemple 11 l'absence de setters, et les champs final.

## Les problèmes d'un modèle unique

Avec un modèle unique les classes du modèle peuvent avoir des dépendances vers différents frameworks ou librairies techniques. Le modèle peut rassembler un mélange de dépendances front, et de dépendances bases de données dans une même classe ce qui ne respecte pas le principe SOLID "Single Responsability Principle". 12

Ici le problème est qu'il y a dans la même

```
import play.data.validation.Constraints;

public class Product {

    @Constraints.Required(message = "This is a mandatory field")
    @Constraints.MaxLength(value = 22, message = "The max length is " + 22)
    @Constraints.Pattern(value = "[a-zA-Z0-9]+", message = "This field is alphanumeric")
    private String ean;

    @Constraints.Required(message = "This is a mandatory field")
    @Constraints.MaxLength(value = 22, message = "The max length is " + 22)
    @Constraints.Pattern(value = "[a-zA-Z0-9]+", message = "This field is alphanumeric")
    private String name;

    @Constraints.MaxLength(value = 150, message = "The max length is " + 150)
    private String description;
```

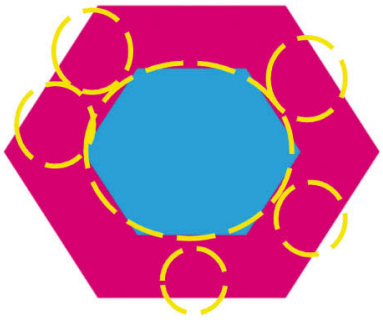
9 *Quelques champs d'une classe du modèle qui portent des annotations de validation.*

```
package fr.catalog.infra.dao;

import fr.catalog.business.Product;
import io.ebean.Finder;
import io.ebean.Model;

import javax.persistence.Entity;
import javax.persistence.Id;
import javax.validation.constraints.Size;
import java.util.List;
import java.util.Optional;

@Entity
public class ProductEntity extends Model {

    @Id
    public String ean;

    public String name;
```

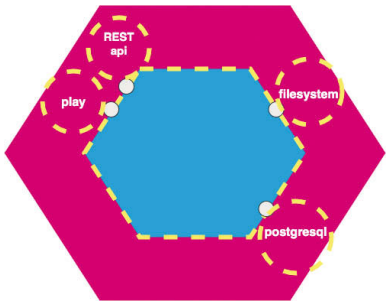
10 *Quelques champs d'une entity avec annotations et super classe pour mapper l'objet à la base.*

```
package fr.catalog.business;

public class Product {
    private final String ean;
    private final String name;
    private final String description;
    private final byte[] picture;

    private Product(String ean, String name, String description, byte[] picture) {
        this.ean = ean;
        this.name = name;
        this.description = description;
        this.picture = picture;
    }

    public static Product newProduct(String ean, String name, String description, byte[] picture) {
        return new Product(ean, name, description, picture);
    }

    public String getEan() { return ean; }

    public String getName() { return name; }

    public String getDescription() { return description; }

    public byte[] getPicture() { return picture; }
}
```

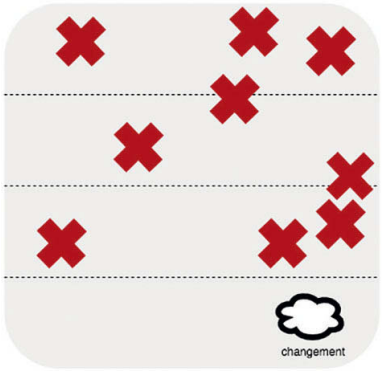
11 *Exemple de classe du modèle business.*



13 *Petites zones d'impact.*



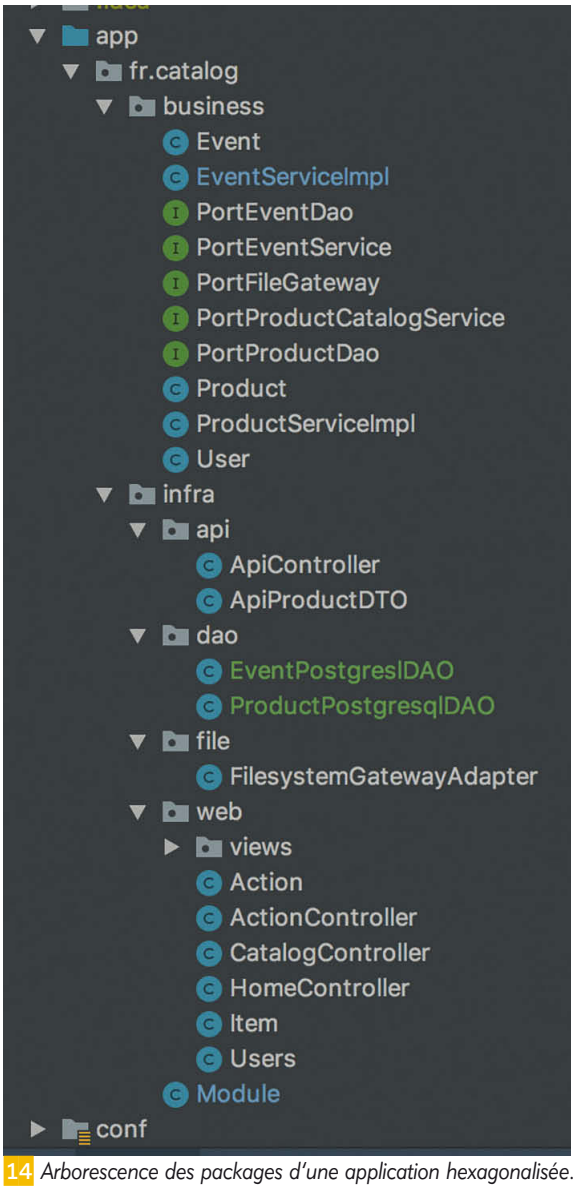

15 *Application hexagonalisée : petites zones d'impacts*




16 *Application fortement couplée : 1 changement, beaucoup d'impacts*

```
package models;

import io.ebean.Finder;
import io.ebean.Model;
import play.data.validation.Constraints;

import javax.persistence.Entity;
import javax.persistence.Id;
import javax.validation.constraints.Size;
import java.util.List;
import java.util.Optional;

@Entity
public class ProductEntity extends Model {

    @Id
    @Constraints.Required
    public String ean;

    @Constraints.Required
    public String name;
```

12 *Exemple d'une classe du modèle unique d'une application.*




14 *Arborescence des packages d'une application hexagonalisée.*

classe des dépendances de librairies front et web avec des librairies dont la responsabilité est de s'occuper de la base de données. Avec un modèle unique, on crée du couplage fort de toutes les couches de l'application.

### Petites zones d'impacts

En découplant l'application, on construit de petites zones d'impact. Chaque changement apporté à l'application comporte un risque moindre, et un gain de temps. 13

### Séparation en packages 14

Séparer l'application en différentes parties peut se faire par des packages. Il y a les deux blocks business et infra eux même redécoupés.

Le modèle de données, les ports et les implémentations des services se trouvent dans le package business.

L'infra est composée des parties *api*, *dao*, *file*, et *web* (package) avec les controllers et templates HTML. Chacune des parties utilisent son propre modèle, qui sera converti dans le modèle métier pour les interactions avec le business. Pour un découplage encore plus fort, on pourrait faire une **séparation en modules**, qui sera mise en place avec l'outil de build utilisé.

### Cas d'une migration technique PostgreSQL vers MongoDB

Prenons le cas d'une migration de postgreSQL vers mongoDB. Il suffit de supprimer le contenu du package DAO, et de le remplacer par les classes accédant au mongoDB. Le risque va se trouver au niveau de la transformation du modèle d'entity mongoDB vers le modèle business. Qui dans le cadre d'un migration technique n'a aucune raison de changer. 15

Dans le cas d'une application fortement couplée, toutes les couches seront impactées augmentant le risque de régression. Pour un changement, les impacts sont nombreux. 16

### Avantages et inconvenients

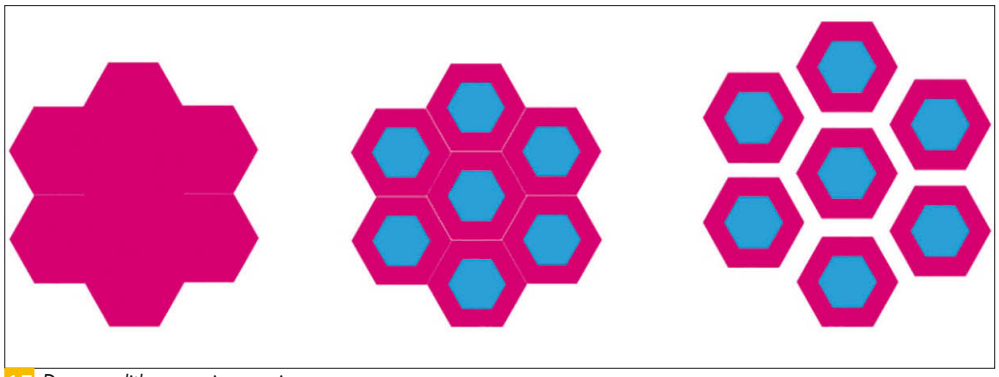
L'inconvénient de ce style de design est évidemment le temps de sa mise en place. Ce découplage a un coût, développer un modèle spécifique pour chaque parties de l'application n'est pas neutre.

Pour les applications CRUD (Create Read Update Delete), avec peu de règles de gestion ce n'est pas nécessaire de mettre en place une telle architecture.

Par contre dans les applications avec du métier les avantages sont nombreux.

Pendant la phase de développement, les changements viennent principalement du Product Owner, ou de feedbacks des utilisateurs, ou encore des développeurs qui challengent l'application tout au long de l'avancement des développements. Assez rapidement on finit par compenser le temps passé sur la mise en place en absorbant les changements continuels rapidement.

Quelques temps après la mise en production, le besoin est figé. Les changements à venir sur les applications



17 *Du monolithe au microservices*

seront des changements techniques comme des migrations ou montées de versions. Le bénéfice du découplage de l'architecture hexagonale simplifiera la mise en oeuvre de ces tâches.

Ou encore de refactoring pour des besoins d'urbanisation du SI.

Pour résumer, voici une liste des avantages :

- Gain de temps lors des changements du métier
- Gain de temps lors des migration techniques
- Intégration facile avec n'importe quel composant ou autre application du SI
- Agilité
- Testablité

## Conclusion

L'architecture hexagonale peut être un bon moyen de séparer une application monolithique en microservices. Chaque microservice étant un hexagone. 17

Il n'y a pas un seul choix d'architecture valable s'appliquant à toutes les entreprises. Si vous envisagez de faire évoluer le système d'information vers des microservices mais que votre système d'information n'est pas encore prêt pour cela, vous pouvez développez un monolithe en le découpant en plusieurs hexagones, cela peut être un bon compromis pour ensuite extraire un ou plusieurs de ces hexagones en microservices.

Le cas du monolithe legacy est intéressant. Après avoir défini les *bounded contexts* pour permettre un découpage intelligent en microservices, vous pouvez identifier les briques à extraire en premier lieu et prioriser le refactoring du monolithe. Refactorer tout le monolithe serait titanesque, voir impossible. On pourrait par exemple se concentrer sur l'extraction d'un microservice de communication transverse au SI (envoi d'email, sms, etc).

L'architecture hexagonale ne s'applique pas aux applications pure CRUD avec pas ou peu de règles de gestion mais on en tire des bénéfices dans tous les autres cas.

**Lætitia Avrot** *est experte PostgreSQL chez EnterpriseDB. Elle est co-fondatrice du mouvement Postgres Women, membre du Postgres Advocacy Group et membre du Postgres Funds Group. Elle a aussi été pendant un an membre du comité du Code de Conduite de la communauté PostgreSQL.Elle a écrit plusieurs patchs pour le projet PostgreSQL et donne régulièrement des conférences dans des événements communautaires.*

Partie 2

# Le SQL avancé pour de vraies performances

***Le SQL est souvent sous-estimé par les développeurs car malheureusement peu enseigné dans le supérieur, que ce soit en France ou à l'étranger... Dans cet article, nous allons découvrir trois fonctionnalités très avancées du SQL : les jointures latérales, les window fonctions et les CTE récursives. Les performances de ces 3 fonctionnalités seront comparées aux performances d'un code équivalent en python.***

## LES WINDOW FUNCTIONS

Les window functions permettent de calculer des résultats de fonctions d'agrégat sur un ensemble de tuples (comme le permettrait un group by) sans pour autant obliger les lignes à être groupées dans le résultat final.

### Premier exemple simple

Si nous reprenons l'exemple du country club, on pourrait vouloir voir pour chaque membre du club son nom, son prénom et le nombre total des membres du club. Il n'est pas possible d'obtenir cette information simplement avec une clause group by sans passer par une sous-requête.
Voici la requête avec une sous-requête :

```
select
 firstname,
 surname,
 (
   select count(*)
   from cd.members
 )

from cd.members
```

Cette requête va fournir exactement le même résultat que la requête avec la window function. Les window functions s'écrivent en SQL avec la clause over dans la clause select. Dans cette clause over, on va pouvoir préciser comment on souhaite grouper les données. Dans notre cas, il s'agit de compter la totalité des lignes de la table, donc un group by vide (qui est la plupart du temps omis en SQL). Ma clause over est donc vide.

```
select
 firstname,
 surname,
 count(*) over()

from cd.members
```

En termes de temps d'exécution, mon volume de données est tellement faible qu'il n'y a pas de différence entre les 2 requêtes (entre 0.300 et 0.400 ms), mais la lisibilité de la deuxième requête semble bien plus importante.
Voyons maintenant comment s'en sort Python, en supposant qu'un développeur moyen ignore qu'il peut utiliser une sous-requête scalaire dans la clause select ou que les window functions lui permette d'avoir cette information d'un seul coup.

```
#! /usr/bin/env python3

import sys
import psycopg2

CONNSTRING = "service=pgexercises"

def list_members_and_count():
 "Fetch the list of members and the total number of members"

 sql = """
   select
   mem.firstname,
   mem.surname
   from cd.members as mem
 """
 pgconn = psycopg2.connect(CONNSTRING
 cursor = pgconn.cursor()
 cursor.execute(sql)

 members = cursor.fetchall()
 count = len(members)

 for (firstname, surname) in members:

   print('{0}, {1}, {2}'.format(firstname, surname, count))

if __name__ == '__main__':
 list_members_and_count()
```

Le programme est relativement simple. Il n'est pas nécessaire d'aller demander à la base de données le nombre de ligne de la table car python peut calculer la taille d'une liste très facilement, ce qui permet d'éviter un aller-retour vers la base.
En termes de performance, ce programme s'exécute en approximativement 120 ms alors que mes requêtes SQL mettent entre 0.300 et 0.400 ms. Ce qui fait que le programme python est à peu près 300 fois plus lent que la requête SQL.

### Allons plus loin

Nous voudrions maintenant obtenir la liste des membres du club ainsi que, pour chaque membre, la durée totale de réservation des

équipements et leur rang dans un classement, le membre ayant réservé le plus grand nombre d'heures apparaissant en premier.
Pour pouvoir écrire cette requête, il y a deux choses à savoir :

- La colonne slots de la table bookings contient le nombre de demie-heures de réservation d'un équipement par un membre du club pour un timestamp donné
- La fonction d'agrégation rank permet de donner le rang d'un tuple en fonction des critères d'agrégation et de tri fournis

```
select
 firstname,
 surname,
 /* Sum the number of half hours the equipment was booked by that member*/
 sum(bks.slots)*.5 as "hours",
 /* get the rank */
 rank() over (order by sum(bks.slots) desc) as rank

from cd.bookings bks
 inner join cd.members mems
 on bks.memid = mems.memid

group by mems.memid
order by rank, surname, firstname;
```

Nous avons besoin de la clause group by car nous utilisons la fonction d'agrégation sum. La window function nous permet de récupérer le rang de chaque ligne suivant l'ordre défini dans la clause over.
Comme la clause select (donc, la window function qu'elle contient) est gérée avant la clause order by, il est possible d'utiliser l'alias de cette colonne dans la clause order by, ce qui est très confortable.
Voyons maintenant comment récupérer cette information en Python.

```
#! /usr/bin/env python3

import sys
import psycopg2

CONNSTRING = "service=pgexercises"

def list_members_hours_and_rank():
 "Fetch the list of members, the total number of hours they booked an equipment and their rank"

 sql = """
   select
   firstname,
   surname,
   sum(bks.slots)*.5 as "hours"

   from cd.bookings bks
   inner join cd.members mems
   on bks.memid = mems.memid
   group by mems.memid
   order by hours desc, surname, firstname;
 """

 pgconn = psycopg2.connect(CONNSTRING)
 cursor = pgconn.cursor()
 cursor.execute(sql)

 members = cursor.fetchall()
 i = 1

 for (firstname, surname, hours) in members:

   print('{0}, {1}, {2}, {3}'.format(firstname, surname, hours, i))
   i += 1

if __name__ == '__main__':
 list_members_hours_and_rank()
```

Python peut de lui-même nous calculer le rang de chaque tuple, il est cependant nécessaire de modifier la clause order by pour obtenir les lignes dans le bon ordre. En termes de performance, notre requête SQL s'exécute en à peu près 2 ms, alors que le même programme en python prend 70 ms, ce qui est à peu près 35 fois plus lent.

## Contre-exemple

On pourrait être tenté de ne plus utiliser que des window functions en lieu et place de la clause group by pour nos fonctions d'agrégat.
Prenons un exemple simple du comptage des lignes d'une table. Comme les tables de ma base pédagogique ne contiennent pas assez de lignes, nous allons en générer avec la fonction generate_series.

```
select
 count(*)
from generate_series(1, 10000000) ;
```

Cette requête toute simple peut également s'écrire avec une window function. Cependant, il faudra rajouter une clause distinct, sans quoi nous obtiendrons autant de lignes (identiques) qu'il y a de lignes dans la table des membres.

```
select distinct
 count(*) over()
from generate_series(1, 10000000) ;
```

Sur mon ordinateur, la requête d'agrégat sans window function s'exécute presque 4 fois plus vite. De plus, on remarque que la première requête est plus facile à lire et donc à maintenir. Là encore, mon but était de vous inciter à la prudence : les window functions ne sont pas la panacée des fonctions d'agrégat et il convient de bien se demander si elles sont vraiment nécessaires lorsqu'on en ajoute.

## D'autres fonctions d'agrégat

Il existe plusieurs fonctions d'agrégat que vous pouvez utiliser avec les window functions. La liste des fonctions d'agrégation disponibles pour PostgreSQL se trouve dans la documentation. Parmi celles-ci, je vous conseille d'étudier tout particulièrement les fonctions lead et lag qui permettent de comparer les valeurs d'une ligne avec celle qui la précède et celle qui la suit. Pour utiliser ces fonctions, n'oubliez pas d'utiliser des clauses order by car le SQL ne garantit pas l'ordre dans lequel vous récupérerez vos données sans order by.

**Suite et fin dans le numéro 238.**

**Aurélie Vache**
*Cloud Dev(Ops) chez Continental*
*Duchess France & DevFest Toulouse Leader*
*Toulouse Data Science core-team &*
*Marraine Elles bougent*
@aurelievache

Partie 2

# Les tests d'intégration facile avec Venom !

*Comme vous le savez, "le gras c'est la vie" mais lorsque l'on travaille dans le développement d'applications/et de site web, et ben le diction "les tests c'est la vie" se vérifie également et nous allons voir dans cet article qu'il n'y a pas que les tests unitaires et que créer et automatiser des tests d'intégration cela peut être un jeu d'enfant ;-).*

## Bonus track

### *Exécution des tests d'intégration automatiquement*

Lorsque l'on écrit ses tests d'intégration, très rapidement vient le besoin d'exécution automatique de ces derniers, sur plusieurs environnement différent. Est-ce possible ? J'ai une bonne nouvelle pour vous, la réponse est oui. Vous pouvez, créer une image Docker contenant Venom, utilisez cette dernière dans votre chaîne de CI/CD, et selon l'environnement sur lequel vous désirez exécuter vos tests, il vous suffit de passer vos paramètres de votre outil de CI : Jenkins/Gitlab/CircleCI…, jusqu'à l'exécution des tests via Venom.

Comment est-ce possible ? Grâce à la possibilité de passer des variables/des paramètres à notre suite de test lors de l'exécution de cette dernière, grâce à la définition d'une variable dans la testsuite et à l'utilisation du paramètre **--var** lors du *venom run* :

testsuite.yml :

```
name: IntegrityTest
version: "2"

vars:
 url: "https://monurl.dev/monendpoint"

testcases:

- name: CreateUser
 steps:
 - type: http
  method: POST
  headers:
    Accept: application/json
    Content-Type: application/json
  url: "{{url}}/v1/user"
  retry: 1
  delay: 2
  assertions:
  - result.statuscode ShouldEqual 201
  - result.bodyjson ShouldContainKey id
  extracts:
   result.bodyjson.id: '{{license=.+}}'
```

```
$ venom run matestsuite.yml --var url=http://monurl.dev/monendpoint
```

L'exemple ci-dessus prend en paramètre une url, avec une valeur par défaut, et test un appel HTTP avec la méthode POST vers l'url **http://monurl.dev/monendpoint/v1/user**, en passant dans les headers un Accept et un Content-Type et en validant si le code de réponse HTTP est un code 201 ET que dans la réponse, au format JSON, il y a un élément nommé "id".

### *Export du report au format xUnit*

Venom permet un export du résultat sous forme de rapport au format xUnit :

```
$ venom run sample.yml --format=xml --output-dir="test-results"
SUCCESS sample.yml                    17.002801ms

Total:2 Duration:18.421452ms
OK:2
KO:0
Skipped:0
TestSuite:1
TestCase:2

File test-results/test_results.xml is written


$ cat test-results/test_results.xml
<?xml version="1.0" encoding="utf-8"?><testsuites>
 <testsuite name="Title of TestSuite [sample.yml]" package="sample.yml" tests="2">
  <testcase name="TestCase with default value, exec cmd. Check if exit code != 1">
   <system-out><![CDATA[foo]]></system-out>
   <system-err></system-err>
  </testcase>
  <testcase name="Title of First TestCase">
   <system-out><![CDATA[foobar]]></system-out>
   <system-err></system-err>
  </testcase>
 </testsuite>
</testsuites>%
```

Je trouve cet export super pratique comme step à rajouter dans son pipeline de sa chaîne de CI/CD.

### *Exécution de plusieurs suite de tests*

Si vous souhaitez exécuter plusieurs suite de test, sans avoir à lancer plusieurs "venom run" c'est possible :

```
$ venom run *
```

Ou bien si vos yaml sont dans un répertoire "tests" :

```
$ venom run tests/*
SUCCESS tests/dogs.yml                1.1743865s
FAILURE tests/cats.yml                2.2076141s
```

## Conclusion

Venom, le petit outil par OVH, est très pratique pour écrire et exécuter ses tests d'intégration. Le seul bémol est qu'il n'est pas assez connu, qu'il manque encore de documentation et de tutoriels assez poussé et qu'il ne supporte pas encore le mutual TLS. De ce fait, parfois, le plus simple est d'écrire une issue sur Github afin d'obtenir une réponse assez rapide pour résoudre notre problème. De mon point de vue, je recommande l'utilisation de cet utilitaire mais j'aimerai une meilleure documentation :-).

**Gaëlle Acas**
*Site Reliability Engineer chez Talend*
**co-organisatrice du CNCF (Cloud Native Computing Foundation) Meetup Nantes.**
*Cloud Addict, j'adore jouer avec les containers, surfer sur les skills Dev&Ops & jongler dans le monde du Serverless*

# Contour, un proxy pour Kubernetes

Partie 2

*Aujourd'hui, il existe de multiples solutions de proxy dans le monde de Kubernetes (nginx, traefik ...). Dans cet article nous allons explorer le projet Contour de VMware.*

## 5 - Les autres pouvoirs de Contour

Nous avons vu l'intérêt de Contour dans le n°236 dans un contexte multi-team avec les ressources HTTPProxy. Notre Team "Santa Ho Ho Ho" peut maintenant gérer les NDD et déléguer la gestion des routes à ses équipes Lego et Playmo.

### A - Gestion des certificats TLS (HTTPS)

Nous étions jusqu'à présent en HTTP, il nous faut maintenant passer en HTTPS afin de sécuriser les requêtes. Il existe des projets Open-Source permettant de générer des certificats TLS automatiquement tel que cert-manager (via letsencrypt) mais pour faire plus simple nous allons créer notre certificat à la main et le stocker sur notre cluster GKE (cf tuto pour OSX https://hackmd.io/@khaly/S1hQGe414):

#### Création du certificat

*création de notre certificat pour le domaine* santa.local :

```
$ openssl req -x509 -out santa.crt -keyout santa.key \
 -newkey rsa:2048 -nodes -sha256 \
 -subj '/CN=santa.local' -extensions EXT -config <( \
  printf "[dn]\nCN=santa.local\n[req]\ndistinguished_name =
dn\n[EXT]\nsubjectAltName=DNS:santa.local\nkeyUsage=digitalSignature\nextendedKe
yUsage=serverAuth")
```

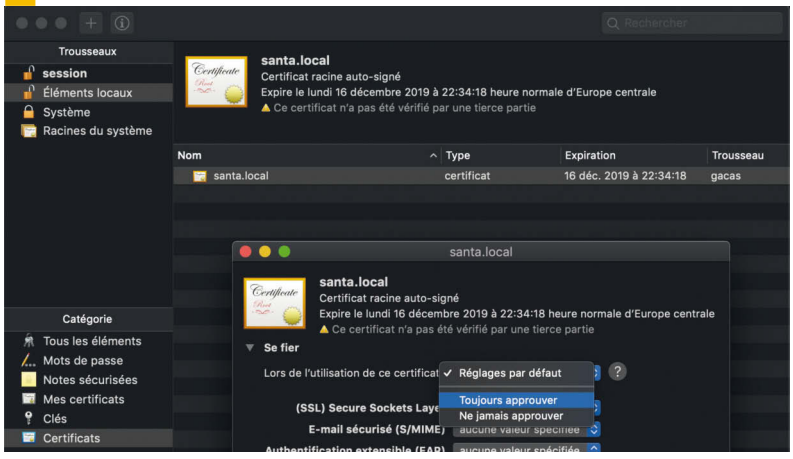
Il faut ensuite l'ajouter au trousseau en local. Je suis sur un Mac-Book ce sera donc via l'utilitaire */**Applications**/**Utilitaires**/**Trousseau d'Accès***. Glissez et déposez le fichier **santa.cert** précédemment généré, double-cliquez dessus et sélectionnez : **"Toujours approuver"**. *Ajout du certificat au trousseau* : 2

#### Configuration de l'Ingress

Pour associer un certificat à un Ingress il faut d'abord le stocker dans une ressource **secret** :

*Ajout du certificat en tant que secret* **santa-tls** *dans le namespace* **santa** :

2




```
$ kubectl create secret tls santa-tls  --cert santa.crt --key santa.key -n santa
```

On peut maintenant configurer notre ressource HTTPProxy root-santa :

```
apiVersion: projectcontour.io/v1
kind: HTTPProxy
metadata:
 name: root-santa
 namespace: santa
 labels:
  app: root-santa
spec:
 virtualhost:
  fqdn: santa.local
  # On ajoute ici le secret contenant le certificat TLS
  tls:
   secretName: santa-tls
 includes:
```

.....Santa Ho Ho Ho est à présent sécurisé : https://santa.local 3

Jusqu'ici nous n'avons rien vu de bien nouveau. Là où cela devient intéressant c'est quand la team des Playmo souhaite avoir un sous-domaine playmo.santa.local ! Santa tient à gérer lui même tous les certificats, pour ce faire nous allons utiliser la délégation des certificats.

#### Délégations des certificats

Il faut donc créer un **wildcard** certificat ***.santa.local** en suivant la même procédure que pour **santa.local** et ensuite le placer dans un secret dans le **namespace santa** :

```
$ kubectl create secret tls wildcard-tls  --cert santa-wildcard.crt --
key santa-wildcard.key -n santa
```

Pour gérer cette délégation, Contour dispose d'une nouvelle ressource, le **TLSCertificateDelegation** :

SSLDelegation.yaml :

```
apiVersion: projectcontour.io/v1
kind: TLSCertificateDelegation
metadata:
 name: santa-wildcard
 namespace: santa
spec:
 delegations:
  - secretName: wildcard-tls
    targetNamespaces:
    - "*"
```

Le "*" signifie ici que le certificat (secret) wildcard-tls sera accessible depuis tous les namespaces du cluster. Appliquer la délégation : kubectl apply -f SSLDelegation.yaml

La team Playmo va donc pouvoir créer sa ressource HTTPProxy (root-playmo) pour le sous domaine playmo.santa.local :

kubectl apply -f playmo-fqdn.yaml

```
apiVersion: projectcontour.io/v1
kind: HTTPProxy
metadata:
 name: root-playmo
 namespace: playmo
spec:
 virtualhost:
  fqdn: playmo.santa.local
  tls:
   secretName: santa/wildcard-tls
 routes:
  - services:
    - name: playmo-v1
      port: 80
```

Playmo peut donc maintenant bénéficier du wildcard et être disponible en HTTPS. Il suffit simplement de pointer sur le secret **santa/wildcard-tls** soit <NAMESPACE_PATH>/<SECRET_NAME> 4

Cette délégation de certificat est vraiment intéressante quand il y a plusieurs équipes. Cela permet de centraliser et de maîtriser les credentials tout en laissant les équipes autonomes.

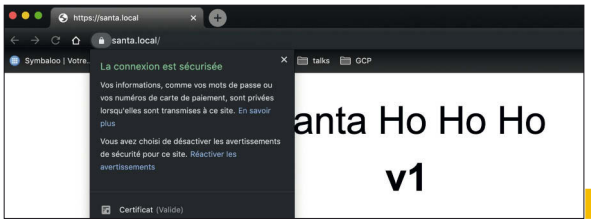
### B - Stratégie de déploiement

Outre l'aspect des multi-team, Contour dispose aussi de fonctionnalités bien pratiques telles que les différentes stratégies de déploiement : blue/green, canary et même du mirroring depuis la v1.0. Prenons un exemple concret où Santa souhaite mettre à jour son site, soit passer à la v2 de Santa Ho Ho Ho. Afin de tester si tout fonctionne bien, il veut passer par un déploiement blue/green. Cela signifie que les 2 versions de son site seront disponibles sur 2 endpoints différents :

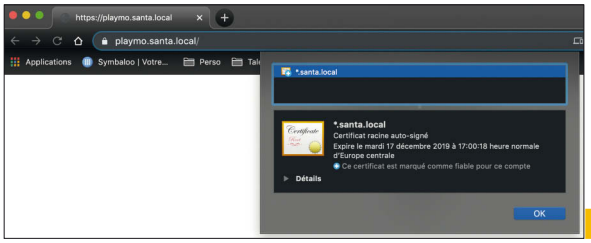
santa-ingress.yaml:

```
apiVersion: projectcontour.io/v1
kind: HTTPProxy
metadata:
 name: root-santa
 namespace: santa
 labels:
  app: root-santa
spec:
 virtualhost:
  fqdn: santa.local
  tls:
   secretName: santa-tls
 # Route pour Santa Ho Ho Ho
 routes:
  - services:
    - name: santa-v1
      port: 80
  - conditions:
    - prefix: /v2
    services:
     - name: santa-v2
       port: 80
```

La v1 est accessible sur l'url de production et sur https://santa.local/v2



3



4

on obtient la v2. Et si on souhaite faire du **canary** c'est aussi simple que le blue/green : santa-ingress.yaml :

```
apiVersion: projectcontour.io/v1
kind: HTTPProxy
metadata:
 name: root-santa
 namespace: santa
 labels:
  app: root-santa
spec:
 virtualhost:
  fqdn: santa.local
  tls:
   secretName: santa-tls
 # Route pour Santa Ho Ho Ho
 routes:
  - services:
    - name: santa-v1
      port: 80
      weight: 10
    - name: santa-v2
      port: 80
      weight: 90
```

Le trafic est réparti de la façon suivante : 10% sur la v1 et 80% sur la v2.

## Conclusion

Contour est idéal pour les organisations en plusieurs équipes. Depuis l'arrivée des micro-services, ce genre de configuration est de plus en plus fréquente. C'est un reverse-proxy qui promet un bel avenir. Contour a pour objectif de devenir un proxy multi-cluster, il a d'ailleurs été créé pour le projet Gimbal (https://github.com/vmware-tanzu/gimbal), un load balancer permettant de gérer le trafic sur plusieurs clusters tels que Kubernetes ou OpenStack.

J'aurai aimé vous en dire plus sur Contour mais le mieux est de le voir par vous même, la documentation est très bien faite : https://projectcontour.io/docs/v1.0.0/

Vous pouvez retrouver les ressources utilisées dans cet article sur mon github: 0https://github.com/gaelleacas/contour-demo

Et si vous avez utilisé un cluster GKE, n'oubliez pas de le supprimer après vos tests. Executez la commande suivante à l'aide du SDK google (gcloud). Suppression du cluster contour-demo:

```
$ gcloud container clusters delete contour-demo
```

Duy Anh PHAM
Clevermind

# L'essentiel des threads pour comprendre les phénomènes de concurrence

Partie 1

*Avec la généralisation des processeurs multicœurs dans les appareils tels que les ordinateurs, smartphones, tablettes et autres dispositifs avec système d'exploitation, développer une application multithreadée devient un enjeu majeur pour répondre aux besoins toujours croissants et nombreux des utilisateurs.*

Les threads sont classiquement utilisés dans les IHM sur des applications de type client lourd ou web pour une expérience utilisateur fluide. Mais ils sont également utilisés dans les applications en backend notamment pour des traitements de type batch ou pour traiter en masse des données ou flux continus en parallélisant les tâches.

Même si développer des applications multithreadées devient plus facile avec l'utilisation de librairies spécialisées ou de frameworks, celles-ci peuvent contenir des bugs dont la recherche de la cause peut être particulièrement difficile. Ce dossier propose de comprendre les concepts fondamentaux des threads, les phénomènes et erreurs rencontrés illustrés avec des programmes en Java.

## Thread et processus

Un processus est l'exécution d'un programme avec son propre espace mémoire (incluant les segments de code, données et piles), entrées/sorties et temps CPU alloués par le système. Cela signifie que deux processus se lancent dans deux espaces mémoire distincts. Les processus peuvent communiquer entre eux avec les mécanismes IPC (Inter Process Communication), avec des pipes ou sockets. Le lancement d'un programme Java est donc un processus.

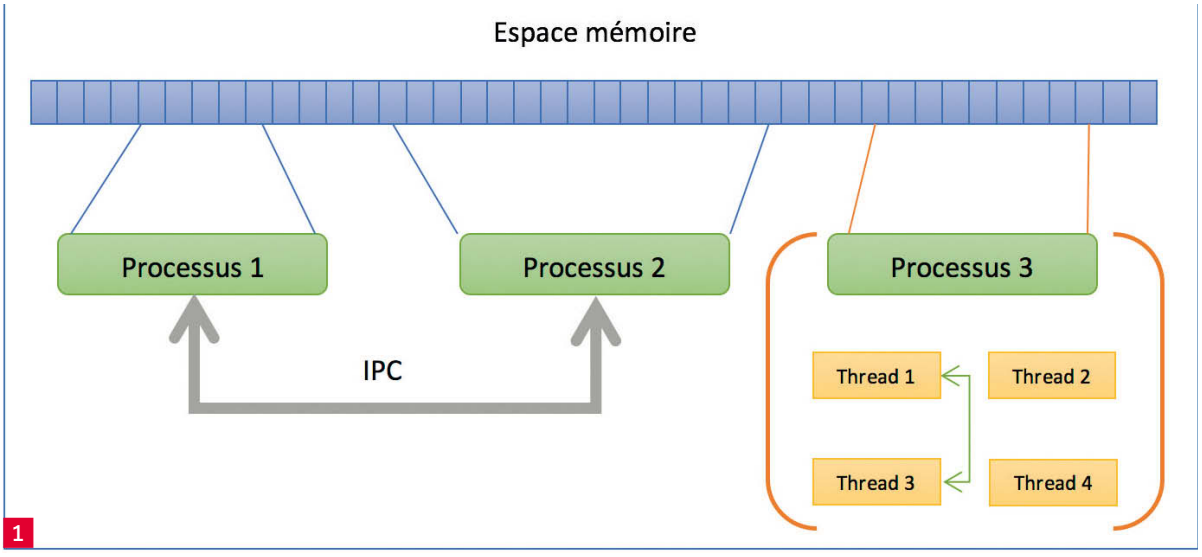
Un thread est un fil d'exécution d'une portion de programme qui s'exécute à l'intérieur même d'un processus. Plusieurs threads peuvent exister au sein d'un processus et chaque processus possède au moins un thread pour être exécuté. Il est qualifié de processus léger, car d'une part la création d'un thread est moins coûteuse que celle d'un processus et d'autre part les threads d'un même processus se partagent sa mémoire virtuelle. Ils possèdent néanmoins possèdent leur propre pile d'appel. Enfin les threads peuvent communiquer directement entre eux. 1

## Lancement d'un thread

Un thread en Java est une instance de la classe java.lang.Thread. Deux manières existent pour qu'un programme soit exécuté par un thread :

- étendre de la classe Thread et surcharger la méthode run()

```
public class MyThread {

    public static void main(String[] args) {

        Thread t = new Thread() {
            @Override
            public void run() {
                System.out.println("thread started");
                for (int i = 0; i < 10; i++) {
                    System.out.println("thread working : " + i);
                }
                System.out.println("thread stopped");
            }
```




1

```
        };

        t.start();
    }

}
```

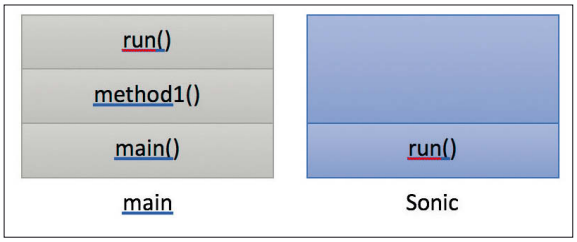
- implémenter la méthode run() de l'interface java.lang.Runnable et l'associer à une instance de Thread (style Java 8 avec les lambda)

```
public class MyRunnable {

    public static void main(String[] args) {

        Runnable r = () -> {
            System.out.println("thread started");
            for (int i = 0; i < 10; i++) {
                System.out.println("thread working : " + i);
            }
            System.out.println("thread stopped");
        };

        Thread t = new Thread(r);
        t.start();

    }

}
```

Il est préférable d'implémenter l'interface Runnable plutôt que d'étendre de Thread pour une meilleure conception orientée objet. En effet, en Java il n'est pas possible d'hériter de plus d'une classe.
L'appel à la méthode run() ne démarre pas par un nouveau thread. Son exécution se lance dans le thread courant.
À noter que tout programme Java se lance dans un processus appelé la JVM avec au minimum un thread principal qui a pour nom main.
Pour pouvoir déclencher réellement un thread, c'est-à-dire un nouveau fil d'exécution, c'est la méthode start() qui doit être invoquée. C'est elle qui se chargera d'invoquer la méthode run() où réside le code de la tâche à accomplir.

**Piles d'appel avec 2 threads**

```
public class StackMain {

    public static void main(String[] args) {
        method1();
        System.out.println(Thread.currentThread() + " : End main");
    }

    private static void method1() {

        Runnable r = () -> {
            System.out.println(Thread.currentThread() + " : task running");
        };

        Thread t = new Thread(r, "Sonic");
        t.run();
        t.start();
    }

}
```




Les threads main et Sonic sont en concurrence avec le résultat suivant :

```
Thread[main,5,main] : task running
Thread[main,5,main] : End main
Thread[Sonic,5,main] : task running
```

La méthode static **Thread.currentThread()** renvoie le thread courant en cours d'exécution. La méthode toString() renvoie la concaténation du nom du thread, la priorité et le nom du groupe.
Le groupe est celui du parent qui lui a donné naissance. Le thread Sonic est bien issu du thread main.
Le résultat illustre bien que :
- l'appel à t.run() s'exécute dans le thread main (et non Sonic)
- l'appel à t.start lance un nouveau thread nommé Sonic

Le nombre 5 indique la priorité.

## Priorités, scheduler

Chaque thread possède une priorité qui lui permet d'obtenir une précédence sur un autre thread et donc d'être choisi par le scheduler(1) pour être exécuté. En Java, les niveaux varient de 1 pour le niveau minimum à 10 qui est la plus haute priorité.
Aucune spécification n'impose à la JVM d'utiliser un scheduler avec time slicing. Rien ne garantit que la JVM termine un thread de basse priorité avant un thread prioritaire, car le scheduler dépend du système d'exploitation. Ce dernier peut très bien laisser finir l'exécution d'un thread en cours avant de choisir un autre thread, ou alors donner la priorité au thread qui attend le plus longtemps.

```
public class ThreadPriority {

    public static void main(String[] args) {

        Runnable r = () -> {
            System.out.println(Thread.currentThread().getName() + " thread started");
            for (int i = 0; i < 5; i++) {
                System.out.println(Thread.currentThread().getName() + " thread working : " + i);
            }
```

*(1) L'OS garde généralement la main sur la priorité des processus et donc l'ordre d'exécution. NDLR*

```
            System.out.println(Thread.currentThread().getName() + " thread stopped");
        };

        Thread t1 = new Thread(r, "t1");
        t1.setPriority(2);

        Thread t2 = new Thread(r, "t2");
        t2.setPriority(Thread.MAX_PRIORITY);

        Thread t3 = new Thread(r, "t3");
        t3.setPriority(Thread.MIN_PRIORITY);

        Thread t4 = new Thread(r, "t4");
        t4.setPriority(Thread.NORM_PRIORITY);

        t1.start();
        t2.start();
        t3.start();
        t4.start();
    }
}
```

Un résultat possible d'exécution donne

```
t1 thread started
t4 thread started
t3 thread started
t2 thread started
t3 thread working : 0
t4 thread working : 0
t1 thread working : 0
t1 thread working : 1
t1 thread working : 2
t1 thread working : 3
t1 thread working : 4
t1 thread stopped
t4 thread working : 1
t4 thread working : 2
t4 thread working : 3
t4 thread working : 4
t4 thread stopped
t3 thread working : 1
t2 thread working : 0
t2 thread working : 1
t2 thread working : 2
t3 thread working : 2
t2 thread working : 3
t3 thread working : 3
t2 thread working : 4
t3 thread working : 4
t2 thread stopped
t3 thread stopped
```

Comme le résultat le montre :

- le thread t1 qui a une priorité de 2, se termine tout de même avant les threads 2 et 4 qui ont pourtant une priorité plus importante respectivement de 10 et 5
- le thread t2 ayant la plus grande priorité 10 se termine après les threads 1 et 4 de priorité respective 2 et 5
- le thread t3 qui a la plus basse priorité 1 se bas au coude à coude avec le thread 2 de priorité 10 pourtant. Leurs exécutions sont entrelacées
- le thread 4 de priorité 5 se termine avant tout de même avant le thread 2 censé être prioritaire.

Il en ressort de ce test qu'un thread ayant une grande priorité ne se termine pas forcément en premier. Être prioritaire signifie avoir plus de chance d'être choisi par le scheduler parmi d'autres threads. Ce dernier peut décider à tout moment de choisir un autre thread et de le laisser terminer par exemple.

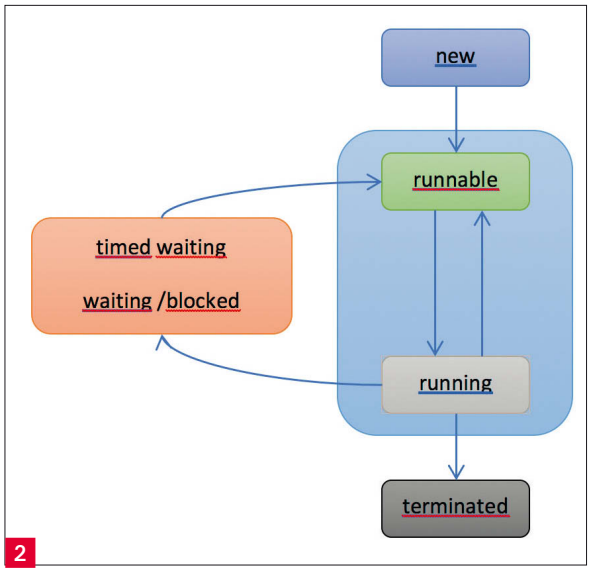
Ne pas spécifier de priorité à un thread équivaut à lui donner 5 comme valeur par défaut (ou Thread.*NORM_PRIORITY*).

## États et transitions

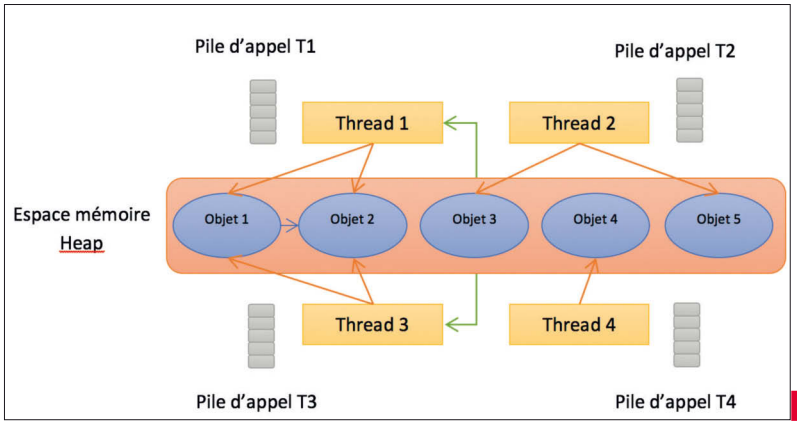
Un thread suit un cycle de vie avec des états bien définis. 2

Les états RUNNABLE et RUNNING sont réunis en un seul état RUNNABLE en examinant la javadoc ou un thread dump (ensemble des stack traces de tous les threads exécutés dans la JVM avec l'état en cours à un instant t).

| État | Description | Transition vers cet état par nom de méthode [classe ] ou mot clé |
|---|---|---|
| New | C'est l'état qui fait suite à l'instanciation d'une classe Thread et avant l'appel à la méthode start() sur le thread qui n'est pas considéré comme alive | new Thread() |
| Runnable | Après t.start(), le thread devient éligible à l'exécution, mais n'est pas choisi par le scheduler. Il est considéré comme alive.Un thread peut revenir à cet état lorsqu'il est running, sleeping, waiting ou blocked | start() [Thread]<br>yield() [Thread] |
| Running | Le thread a été choisi par le scheduler et c'est là que le code de la méthode run() est exécuté par la JVM | run() [Thread] |
| Timed waiting | Le thread attend pendant un temps fini ou qu'un autre thread effectue une action pendant un temps fini.Une fois le temps expiré, le thread redevient runnable | sleep(timeout) [Thread]<br>join(timeout) [Thread]<br>wait(timeout) [Object] |
| Waiting | Le thread attend indéfiniment qu'un autre thread effectue une action. Le thread repassera à l'état runnable lorsque l'autre thread qu'il attend lui enverra une notification par la méthode notify() ou notifyAll() | join() [Thread]<br>wait() [Object]<br>notify() [Object]<br>notifyAll() [Object] |
| Blocked | Le thread attend qu'un verrou de synchronisation se libère pour entrer dans un bloc de code synchronisé | synchronized |
| Terminated | Fin de la méthode run(), le thread est terminé et n'est plus considéré comme alive. Un nouvel appel à t.start() va déclencher une IllegalThreadStateException | N/A |



2



3

## Race condition

Les threads se partagent la mémoire au sein d'un même processus.
3

Les threads 2 et 4 sont indépendants puisqu'ils travaillent avec des objets distincts : objets 3 et 5 pour le thread 2 et l'objet 4 pour le thread 4.
Quant aux threads 1 et 3, ils accèdent à des ressources communes, en l'occurrence les objets 1 et 2 liés entre eux, et peuvent modifier leur état conduisant à un état inconsistant source d'erreur et de comportement arbitraire. Cette situation est désignée par race condition. Il est donc nécessaire de synchroniser l'accès.
Le concept de thread safety désigne la propriété d'un code qui peut être exécuté correctement par plusieurs threads.

### Exemple de code non thread-safe

Le programme ci-dessous montre un couple qui veut retirer tout l'argent sur un compte commun avec un solde de 100 par retrait de 20.

```
public class AccountDemo implements Runnable {

    private Account account = new Account();

    public static void main(String[] args) {

        AccountDemo r = new AccountDemo();

        Thread t1 = new Thread(r, "Alice");
        Thread t2 = new Thread(r, "Bob");
        t1.start();
        t2.start();
    }

    @Override
    public void run() {
        for (int i = 0; i < 5; i++) {
            makeWithdrawal(20);
            if (account.getBalance() < 0) {
                System.out.println("account is overdrawn!");
            }
        }
    }

    private void makeWithdrawal(int amount) {

        if (account.getBalance() >= amount) {
            System.out.println(Thread.currentThread().getName() + " is going to withdraw");

            try {
                Thread.sleep(500);
            } catch (InterruptedException e) {
            }

            account.withdraw(amount);
            System.out.println(Thread.currentThread().getName() + " completes the withdrawal");
        } else {
            System.out.println(
                    "Not enough in account for " + Thread.currentThread().getName() + " to
withdraw " + account.getBalance());
        }
    }

    private class Account {

        private int balance = 100;

        int getBalance() {
            return balance;
        }

        void withdraw(int amount) {
            this.balance = this.balance - amount;
        }
    }

}
```

L'exécution du programme donne un résultat possible suivant (une exécution répétée peut faire varier le résultat)

```
Alice is going to withdraw
Bob is going to withdraw
Bob completes the withdrawal
Bob is going to withdraw
Alice completes the withdrawal
Alice is going to withdraw
Bob completes the withdrawal
Bob is going to withdraw
Alice completes the withdrawal
Alice is going to withdraw
Bob completes the withdrawal
Not enough in account for Bob to withdraw 0
Not enough in account for Bob to withdraw 0
Alice completes the withdrawal
account is overdrawn!
Not enough in account for Alice to withdraw -20
account is overdrawn!
Not enough in account for Alice to withdraw -20
account is overdrawn!
```

Le résultat montre que :

- les retraits mettent le compte à découvert même si le code contient un contrôle sur le solde avant le débit du montant voulu.
- les 2 opérations (contrôle + retrait) peuvent être exécutées par Alice et Bob en même temps, car ils sont en concurrence, ce qui fausse complètement l'opération de retrait.

### Les raisons de l'inconsistance des données

L'opération de mise à jour d'un champ par une nouvelle valeur n'est pas atomique. Elle requiert toujours 3 étapes :

- lecture de la valeur courante
- calculs nécessaires et obtention de la nouvelle valeur
- affectation de la nouvelle valeur à la variable

Or comme nous l'avons vu précédemment, le scheduler peut choisir d'arrêter thread en cours d'exécution pour un autre à tout moment et c'est de là que le problème vient. Pour remédier à l'exemple d'Alice et Bob, le contrôle et retrait doivent être vus comme une seule opération atomique. L'ensemble des opérations doivent s'exécuter entièrement par un seul thread à la fois et sans interruption. Nous ne pouvons garantir que le thread reste à l'état running tout au long de l'opération atomique à cause du scheduler. En revanche, ce que nous pouvons garantir c'est qu'un thread exécutant l'opération atomique puisse changer d'état depuis ou vers l'état running sans qu'aucun autre thread n'accède à cette même ressource partagée avec le mécanisme de synchronisation.

### Synchronisation de code

Le mot clé synchronized est utilisé pour créer du code synchronisé. Le mécanisme utilise en interne un verrou associé à une instance de Object ou Class (représentant une classe Java chargée) ce qui permet à la JVM de garantir que le code synchronisé sera exécuté par un seul thread à la fois.

Un thread souhaitant un accès exclusif à une ressource et ses propriétés doit d'abord acquérir un verrou sur un objet et le libérer quand il en a terminé avec. Un autre thread qui tente de prendre ce même verrou se retrouvera mis en attente (état blocked) jusqu'à la libération du verrou. Tant que l'exécution du code synchronisé n'est pas terminée, le thread qui possède le verrou continue de le détenir même si le thread s'endort en passant de l'état runnable à timed waiting. Il ne le libère donc pas. Pour reprendre l'exemple d'Alice et Bob, l'ajout du mot clé synchronized au niveau de la méthode rendra le programme d'exemple thread-safe.

```
private synchronized void makeWithdrawal(int amount) {

    if (account.getBalance() >= amount) {
        System.out.println(Thread.currentThread().getName() + " is going to withdraw");

        try {
            Thread.sleep(500);
        } catch (InterruptedException e) {
        }

        account.withdraw(amount);
        System.out.println(Thread.currentThread().getName() + " completes the withdrawal");
    } else {
        System.out.println(
                "Not enough in account for " + Thread.currentThread().getName() + " to withdraw "
 + account.getBalance());
    }
}
```

À présent, une personne à la fois peut débiter le compte. Alice et Bob ne s'entremêlent plus et peuvent retirer de l'argent sans risquer d'être à découvert.

```
Alice is going to withdraw
Alice completes the withdrawal
Bob is going to withdraw
Bob completes the withdrawal
Bob is going to withdraw
Bob completes the withdrawal
Bob is going to withdraw
Bob completes the withdrawal
Bob is going to withdraw
Bob completes the withdrawal
Not enough in account for Bob to withdraw 0
Not enough in account for Alice to withdraw 0
Not enough in account for Alice to withdraw 0
Not enough in account for Alice to withdraw 0
Not enough in account for Alice to withdraw 0
```

Voyons les différents types de synchronisation.

### Synchronisation sur une méthode non static

Une telle méthode est dite synchronisée lorsqu'elle est marquée par le mot clé synchronized.

```
public synchronized void doAction() {

    // do something by one thread
}
```

Le thread invoquant la méthode obtient le verrou associé à l'objet courant c'est-à-dire l'instance this. C'est ce type de synchronisation qui est employé dans l'exemple d'Alice et Bob.

### Synchronisation sur un bloc de code avec un objet

Ce type de synchronisation permet de spécifier l'objet qui servira de verrou. L'instance courante de l'objet peut servir de verrou pour protéger des variables d'instance.

```
public void doAction() {

    // do something by multiple threads

    synchronized (this) {
        // critical session executed by one thread only
    }

    // continue do something by multiple threads
}
```

Cas particulier, si la synchronisation est faite au niveau de la méthode comme ci-dessous :

```
public synchronized void doAction() {

    // do something by one thread
}
```

alors elle peut être remplacée par une synchronisation sur un bloc de code utilisant l'instance courante :

```
public void doAction() {
    synchronized (this) {
        // do something by one thread
    }
}
```

De manière générale la synchronisation peut utiliser un objet quelconque comme verrou pour protéger les variables d'instance :

```
private final Object lock = new Object();

public void doAction() {

    // do something by multiple threads

    synchronized (lock) {
        // critical session executed by one thread only
    }

    // continue do something by multiple threads
}
```

Il est possible de synchroniser sur un objet static pour protéger des variables static.

```
private static final Object LOCK = new Object();

public void doAction() {

    // do something by multiple threads

    synchronized (LOCK) {
        // critical session executed by one thread only
    }
    // continue do something by multiple threads
```

### Synchronisation d'une méthode static ou sur un bloc de code avec un objet de type Class

L'ajout du mot clé synchronized permet de protéger l'accès à des variables static. Dans ce cas, le verrou utilisé est l'instance de type Class représentant la classe chargée.

```
private static synchronized void doAction() {

    // do something by one thread
}
```

Si le mot clé synchronized est placé au niveau de la méthode, alors la synchronisation peut être remplacée par une synchronisation sur un bloc de code utilisant toujours l'instance de type Class représentant la classe.

```
private void doAction() {
    synchronized (MyClass.class) {
        // critical session executed by one thread only
    }
}
```

### Synthèse des types de synchronisation

```
public class SynchronizationSummary {

    private final Object lockA = new Object();

    private final Object lockB = new Object();

    private static final Object LOCK_C = new Object();

    public void doAction1() {
        synchronized (lockA) {
            // critical session executed by one thread only
        }
    }

    public void doAction2() {
        synchronized (lockB) {
            // critical session executed by one thread only
        }
    }

    public synchronized void doAction3() {
        // critical session executed by one thread only
    }
```

```
    public static synchronized void doAction4() {
        // critical session executed by one thread only
    }

    public void doAction5() {
        synchronized (LOCK_C) {
            // critical session executed by one thread only
        }
    }

    public void doAction6() {
        synchronized (SynchronizationSummary.class) {
            // critical session executed by one thread only
        }
    }
}
```

Soient x et y, 2 instances de la classe SynchronizationSummary. Voici un résumé des méthodes qui ne peuvent jamais être exécutées au même moment.

| Thread 1 | Thread 2 | Commentaires |
|---|---|---|
| **x.doAction1()** | x.doAction1() | Le bloc de code est synchronisé sur le |
| **x.doAction2()** | x.doAction2() | même verrou par les 2 threads |
| **x.doAction3()** | x.doAction3() | |
| **x.doAction4()** | x.doAction4() | La synchronisation est faite sur la méthode |
| **x.doAction4()** | y.doAction4() | static quelles que soient les instances de SynchronizationSummary utilisées par les 2 threads |
| **x.doAction5()** | x.doAction5() | Le bloc de code est synchronisé sur le |
| **x.doAction5()** | y.doAction5() | même verrou static quelles que soient les instances de SynchronizationSummary utilisées par les 2 threads |
| **x.doAction4()** | x.doAction6() | La synchronisation sur la méthode static utilise comme verrou l'instance de type Class représentant SynchronizationSummary quelles que soient les instances utilisées par les 2 threads |

## Bonnes pratiques

La synchronisation rend un code thread-safe en garantissant que l'ensemble des instructions soient exécutées par un seul thread dans la section critique. Mais elle a un coût et c'est la raison pour laquelle la section critique doit être la plus courte possible en utilisant une synchronisation sur un bloc de code afin d'éviter qu'un thread monopolise une ressource trop longtemps. Il convient donc d'identifier les instructions qui peuvent être exécutées par plusieurs threads simultanément de ceux qui ont besoin d'un accès exclusif à une ressource partagée.

Exemple : si une section dure 100 millisecondes et qu'il y a 50 threads en attente du verrou, la durée totale d'exécution de cette section critique est de 5 secondes. Bien que ce soit un temps négligeable pour un humain, cela représente un temps extrêmement long pour un ordinateur.

Un objet qui est utilisé dans un pool, un String ou une primitive autoboxée en wrapper par exemple, ne doit pas être utilisé comme verrou. En effet, un thread qui prend un verrou sur ce String/wrapper empêcherait un autre thread de prendre un verrou défini par une autre variable, mais pointant sur cette même instance de String/wrapper et d'entrer dans la section critique alors que les programmes n'ont rien en commun. En effet, un pool permet de réutiliser des objets déjà créés, ce qui n'est pas le but recherché avec la synchronisation.

Il est préférable d'utiliser une variable distincte de la référence this, privée et finale comme verrou et de créer un bloc de code synchronisé :

- cela permet de réduire la portée de synchronisation aux instructions qui en ont vraiment besoin.
- finale permet d'éviter qu'un setter ou autre code ne vienne modifier la référence du verrou.
- une autre méthode m2() nécessitant une synchronisation et protégeant ses variables n'est pas obligée d'utiliser le même verrou d'une autre méthode m1() si les variables restent consistantes lors d'une invocation simultanée par 2 threads.

**La suite de ce dossier dans le prochain numéro.**

**Christophe PICHAUD**
*Lead Software Architect chez Infeeny*
christophep@cpixxi.com
www.windowscpp.com

# Windows : faire un menu Démarrer en C++

*Dans cet article, nous allons réaliser une application Windows qui ressemble à un menu Démarrer. Il s'agit de lister les applications et de les afficher pour permettre à l'utilisateur de les lancer.*

Pourquoi une telle idée me direz-vous ? Parce que selon moi, le menu Démarrer de Windows 10 est moche. Et puis connaissant les API de dessin GDI+, je me suis imaginé un menu Démarrer avec des rectangles multi-color.

## Préambule sur le développement graphique Windows

Vous allez me dire OK, mais pour développer sous Windows on fait comment ? Ce n'est pas dur, on va utiliser un Framework graphique que Microsoft utilise : *Microsoft Foundation Classes* (MFC). Pour ceux qui pensent que les MFC sont ringards, je leur évoquerais juste DevExpress, SyncFusion, Telerik et leur dirait que ces Frameworks NET sont équivalents. On fait des graphiques, du Ribbon, des grids et des applications qui ont un look détonnant. Oui les MFC ont 25 ans car c'est le début du C++ chez Microsoft. La première fois que les API Windows étaient encapsulées dans des classes C++. A titre de comparaison, QT est né à peu près à la même époque et QT a toujours le vent en poupe…

## Architecture logicielle

Nous allons réaliser une application SDI avec une fenêtre principale et une Dockable Pane encrée à gauche qui présente les applications sous forme d'arbre (TreeView). 1

Pour récupérer la liste des applications et des groupes, on va utiliser une astuce : itérer dans le répertoire « C:\ProgramData\Microsoft\Windows\Start Menu\Programs ». 2

Dans ce répertoire, on a tout ce qu'il faut. Pour chaque élément, on a :

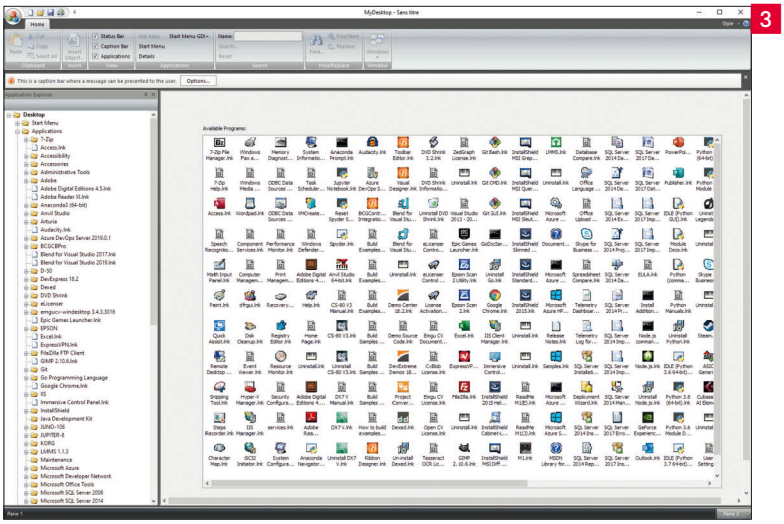
- Un nom
- Un lien
- Une icône

En ce qui concerne l'affichage des applications, on va utiliser différents panneaux (vues):

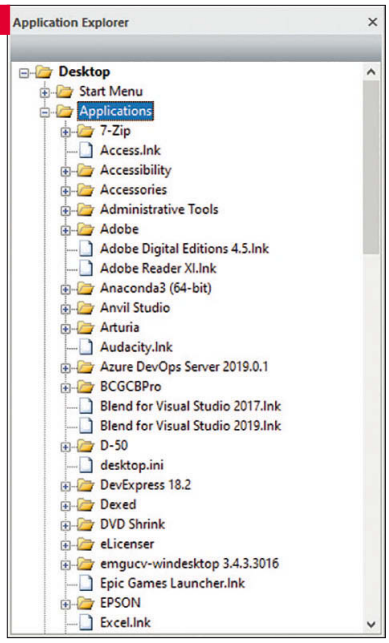
- Une vue pour le détail d'un lien
- Une vue pour un affichage à l'ancienne
- Une vue de type affichage « moderne » via GDI+ 3

## Communication Applicative

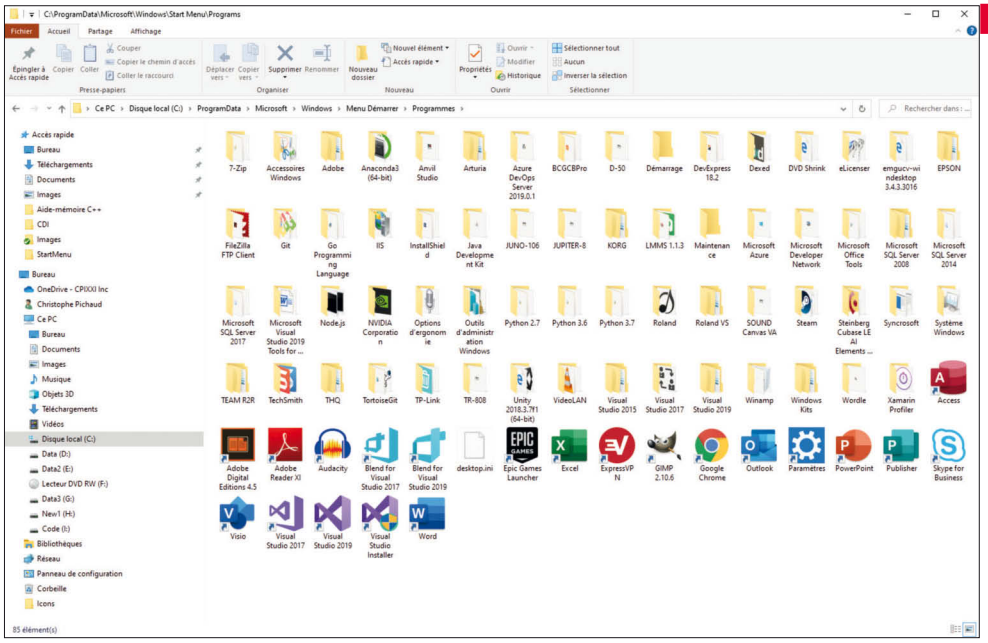
Pour arriver à un résultat final comme celui-ci, il y a des règles à respecter. D'abord, il y a une infrastructure de communication interne

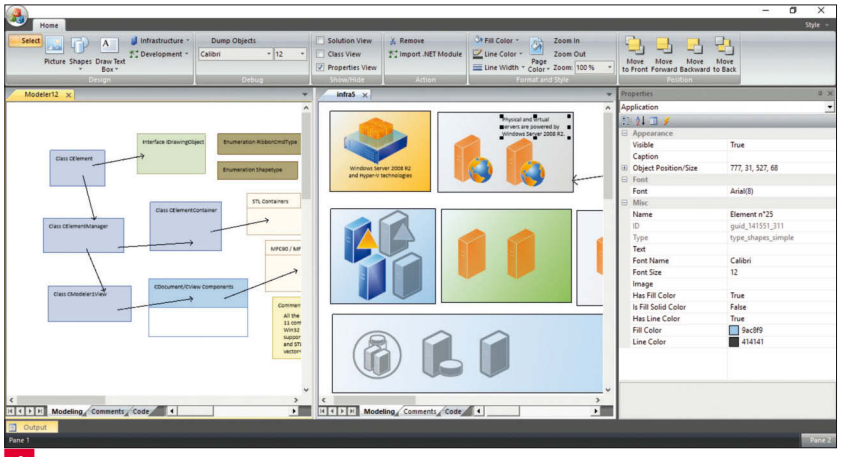

3

1




2





4

à l'application. Je m'explique. C'est une application C++ ; il y a donc des classes et il faut instancier des objets pour que tout fonctionne. Comment ça marche ? L'application contient les classes suivantes :

| Classe | Hérite de | Description |
|---|---|---|
| CMyDesktopApp | CWinAppEx | C'est la classe instanciée en globale |
| CMyDesktopDoc | CDocument | Classe de data à charger ou sauvegarder |
| CMyDesktopViewEx | CFormView | Vue affichée en premier |
| CStartMenuView | CFormView | Vue à l'ancienne |
| CStartMenuViewEx | CScrollView | Vue graphique Moderne |
| CMainFrame | CFrameWndEx | Menu, Ribbon & co |
| CApplicationViewBar | CDockablePane | Panneau de gauche (TreeView) |

## La classe Application et la cascade d'appel du Framework

Comment se lance l'application ? Dans MyDesktop.cpp, il y a une déclaration globale :

```
// The one and only CMyDesktopApp object
CMyDesktopApp theApp;
```

C'est comme ça que tout démarre. C'est le point d'entrée d'une application MFC. A partir de là, les actions suivantes sont enclenchées :

- CMyDesktopApp ::InitInstance
  - Création du triplet <CMainFrame, CMyDesktopDoc, CMydesktopViewEx> comme canvas d'affichage principal
  - Création du Ribbon
- CMainFrame ::OnCreate
  - Création du Ribbon
  - Création du DockablePane avec le TreeView
- CMyDesktopViewEx ::OnInitialUpdate

La symbiose des MFC consiste à créer une interaction entre un MainFrame (la partie visible de l'application, Ribbon, menu, toolbar, dockable panes), un document et une vue principale via cette séquence d'appel dans InitInstance de la classe dérivée de CWInAppEx:

```
// Register the application's document templates.  Document templates
// serve as the connection between documents, frame windows and views
CSingleDocTemplate* pDocTemplate;
pDocTemplate = new CSingleDocTemplate(
    IDR_MAINFRAME,
    RUNTIME_CLASS(CMyDesktopDoc),
    RUNTIME_CLASS(CMainFrame),     // main SDI frame window
    RUNTIME_CLASS(CMyDesktopViewEx));
  //RUNTIME_CLASS(CMyDesktopView));
if (!pDocTemplate)
    return FALSE;
AddDocTemplate(pDocTemplate);
```

C'est la base des MFC pour le modèle Document/Vue.

## Abstraction des MFC

Pour avoir un code lisible, il faut dépasser le cadre des MFC. En effet, le but principal n'est pas de mixer son code au milieu des MFC. C'est trop facile mais ce n'est pas élégant. Ce qu'il faut, c'est y mettre une couche d'abstraction supplémentaire. Dans le cas de cette application, j'y ai introduit la notion de Manager. Un manager est un objet de haut niveau qui connaît tout le monde. Pourquoi une telle idée me direz-vous ? Pour isoler le code et pouvoir le réutiliser.

## Réutilisation d'un canevas de dessin

Pour pouvoir afficher des éléments graphiques dans une vue (CScrollView), j'ai réutilisé l'infrastructure de mon projet UltraFluid Modeler… Voici à quoi ça ressemble : 4

Dans ce projet, un élément graphique :

- Possède des propriétés
- Est sélectionnable, déplaçable, resizable
- Peut passer devant, derrière
- On peut zoomer ou pas

Bref, c'est tout ce dont j'ai besoin, pourquoi le recoder si je peux réutiliser ???

La réalité est un peu plus douloureuse car on tire plusieurs classes mais ce n'est pas grave comparé aux fonctionnalités importées. Voici la liste des classes importées :

| Classe | Hérite de | Description |
|---|---|---|
| CElement | CObject | C'est un objet graphique |
| CElementContainer | CObject | C'est une collection sérialisable d'objets |
| CElementManager | CObject | C'est le Handler global à réutiliser |
| CDrawingContext | | C'est le contexte graphique GDI+ |

## L'assemblage en LEGO

Comment on fait pour merger des classes C++ de plusieurs projets pour que cela marche ? C'est la question à 1 million de dollars ! Il faut se mettre à la place du compilateur. Je dois connaître tous les éléments (token) donc tout doit être déclaré : énumérations, structures, variables globales, types, classes dépendantes, etc.

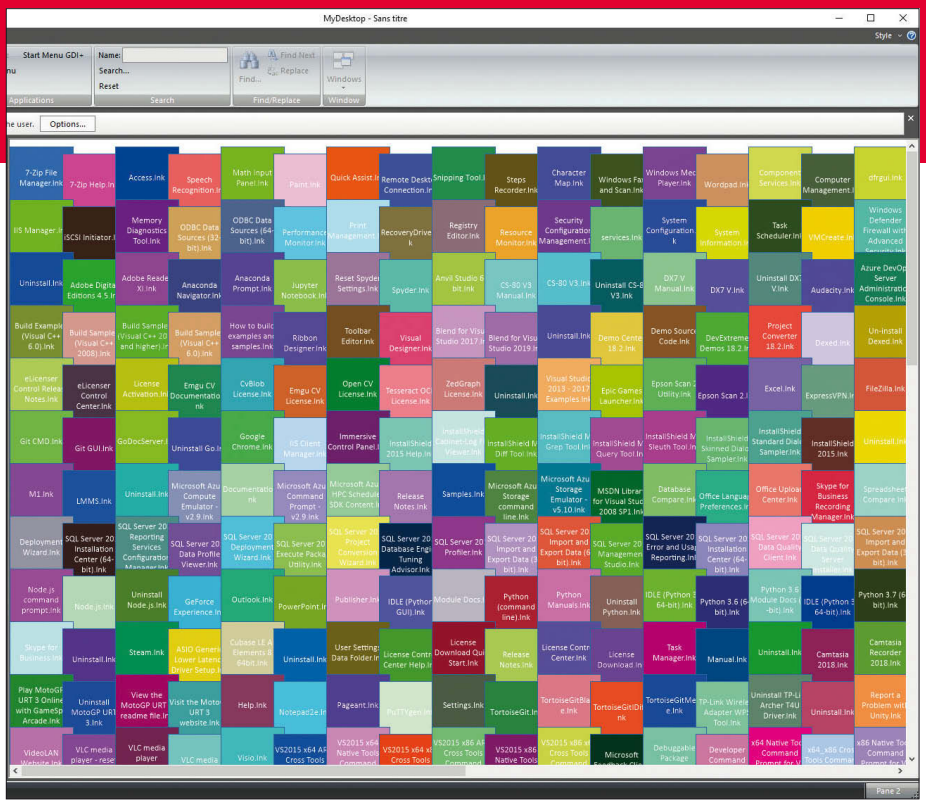
## Le Manager

La classe CElementManager est un handler globale pour une classe dérivée de CScrollView. Une fois que la classe passée en paramètre est bien une classe CStartMenuViewEx, tout est cablé ! Il faut juste créer des éléments graphiques de type CElement et les ajouter au membre m_objects du manager.

La routine principale de l'application est la suivante :

```
GetManager()->LoadStartMenu(this);
```

Au démarrage de l'application, la vue CMyDesktopViewEx va créer la liste des applications et remplir le TreeView du Doackable Pane de gauche via :




```
  CString strPath = _T("C:\\ProgramData\\Microsoft\\Windows\\Start Menu\\Programs");
  CFileManager::SearchDrive(_T("*.*"), strPath, true, false,
pMainFrame->m_wndApplicationView._hAppz);
  pMainFrame->m_wndApplicationView.m_wndFileView.Expand(
pMainFrame->m_wndApplicationView._hAppz, TVE_EXPAND);
```

Voici le code de SearchDrive :

**Code complet sur programmez.com & github**

## Ajout de l'élément graphique

Pour afficher un élément graphique nouveau (un élément de menu démarrer), il faut le définir :

**Code complet sur programmez.com & github**

## Enrichissement de CElementManager

Pour afficher des éléments graphiques, il faut convertir le vector de ApplicationLink en objet CStartMenuElement. C'est le rôle de la méthode LoadStartMenu de la classe CElementManager :

**Code complet sur programmez.com & github**

L'appel à la méthode Invalidate() provoque l'appel à redessiner la vue, donc c'est un appel à OnDraw.

```
void CStartMenuViewEx::OnDraw(CDC* pDC)
{
  CMyDesktopDoc* pDoc = GetDocument();
  ASSERT_VALID(pDoc);
  if (!pDoc)
    return;

  GetManager()->DrawEx(this, pDC);
}
```

Cela se traduit par un appel indirect à Draw une fois que le contexte graphique Windows CDC est obtenu par CElementManager :: Draw(pView, pDrawDC); dans :

```
void CElementManager::Draw(CStartMenuViewEx* pView, CDC* pDC)
{
  // Initialize GDI+ graphics context
  Graphics graphics(pDC->m_hDC);
  // just like that
  //graphics.ScaleTransform(0.75f, 0.75f);
  graphics.ScaleTransform(m_fZoomFactor, m_fZoomFactor);

  // TODO: add draw code for native data here
  for (vector<std::shared_ptr<CElement>>::const_iterator i = GetObjects().begin();
i != GetObjects().end();
i++)
  {
    std::shared_ptr<CElement> pElement = *i;
    pElement->m_pView = pView;

    // Construct the graphic context for each element
    CDrawingContext ctxt(pElement);
    ctxt.m_pGraphics = &graphics;

    //pElement->Draw(pView, pDC);
    pElement->Draw(ctxt);

    if (pView != NULL && pView->m_bActive &&
!pDC->IsPrinting() && IsSelected(pElement))
      pElement->DrawTracker(ctxt, TrackerState::selected);
  }
}
```

Vous allez me dire, OK pour le dessin mais comment on lance les applications ? Il suffit de faire bouton droit, Start.. Il y a un appel à ShellExecute du lien : rien de compliqué.

## Conclusion

La programmation graphique Windows permet de mettre en œuvre toute la puissance de la programmation orientée objet (OOP) et c'est ludique. GDI+ permet toutes les créations graphiques possibles et ce de manière confortable et simple à utiliser. Le code de l'application est disponible sur :

https://github.com/ChristophePichaud/MyDesktop

# Un projet de Noël *

Un jour, il fut décidé de lancer un nouveau projet en suivant la fameuse méthode agile


Que d'innovations ! Les idées fusaient, les sprints s'enchainaient, tout se déroulait à merveille

Mais bientôt, les changements de spécifications se firent quotidiens et le backlog s'emballa

Sans surprise, le client gronda et refusa de décaler la date de mise en ligne


Alors, comme d'habitude, tout le monde hurla, mais tout le monde coda. Et tout fut (plus ou moins) livré le jour J.

CommitStrip.com

Joyeux Noël !

* fonctionne aussi avec Noël 2020.

**Une publication Nefer-IT,** 57 rue de Gisors, 95300 Pontoise - redaction@programmez.com
Tél. : 09 86 73 61 08
Directeur de la publication : François Tonic

**Rédacteur en chef :**
François Tonic

**Ont collaboré à ce numéro :**
ZDnet, CommitStrip

**Nos experts techniques :** C. Pichaud, D. Anh Pham, G. Acas, A. Vaché, L. Avrot, P. Iogna, J. Chomarat, O. Lourme, A. Giretti, V. Michalak, M. Rouillé, T. Leriche, A. Hussin, F. Poindron, V. Fabing, L. Saget-Lethias, P. Prémartin, D. Duplan, V. Loquet, F. Lionet, M. Ellerbach,

**Maquette :** Pierre Sandré.

**Publicité :** François Tonic / Nefer-IT
Tél. : 09 86 73 61 08
ftonic@programmez.com.

**Imprimeur :** SIB Imprimerie

**Marketing et promotion des ventes :**
Agence BOCONSEIL
Analyse Media Etude
Directeur :
Otto BORSCHA
oborscha@boconseilame.fr

**Responsable titre :**
Terry MATTARD Téléphone : 09 67 32 09 34

**Contacts :**
Rédacteur en chef : ftonic@programmez.com
Rédaction : redaction@programmez.com
Webmaster : webmaster@programmez.com
Evenements / agenda :
redaction@programmez.com

**Dépôt légal :** à parution
Commission paritaire : 1220K78366
ISSN : 1627-0908


**Abonnement :**
Service Abonnements PROGRAMMEZ,
4 Rue de Mouchy, 60438 Noailles Cedex.
Tél. : 01 55 56 70 55
*abonnements.programmez @groupe-gli.com*
Fax : 01 55 56 70 91 - du lundi au jeudi de 9h30 à 12h30 et de 13h30 à 17h00, le vendredi de 9h00 à 12h00 et de 14h00 à 16h30.

**Tarifs**
Abonnement (magazine seul) : 1 an - 11 numéros France métropolitaine : 49 € - Etudiant : 39 € CEE et Suisse : 55,82 € - Algérie, Maroc, Tunisie : 59,89 € Canada : 68,36 €
Tom : 83,65 € - Dom : 66,82 €
Autres pays : nous consulter.

**PDF**
35 € (monde entier) souscription sur www.programmez.com

# Actuellement chez votre marchand de journaux

## La nouvelle formule des HS de *Tangente*


**Version numérique sur**
**tangente-mag.com**

**Abonnement ou commande du numéro sur**
**infinimath.com/librairie**

**Maths et emploi en France : cette dynamique récente… s'amplifie !**
**La formation mathématique et informatique, on le savait, est recherchée dans les recrutements des grands groupes. La nouveauté, c'est que les compétences de mathématicien, d'informaticien et d'ingénieur sont aussi demandées dans les PME. C'est évident pour les *startups*, mais toutes les structures constatent que les maths sont un incontournable vecteur d'innovation.**

**Intelligence artificielle, *Data mining*, Modélisation, Simulation, Optimisation, Calcul Haute Performance, Cryptographie, Statistiques et calcul stochastique… sont parmi les thèmes les plus demandés.**

G·ECHO
cybersécurité

## Solutions pour la cybersécurité

MICRO FOCUS
Gold Partner
Security

Sécurité des développements : Fortify
Protection des données : Voltage
Evènements de sécurité : ArcSight

beyond SECURITY

Trouvez vos 0-days : bestorm
Scan de vulnérabilités : besecure

## Conseil, services, développement

- Audits de code, test d'intrusion,
- Conseil, analyse de risque, PSSI,
- R&D, développement de solutions. …

## Formations et recrutement en SSI

**www.g-echo.fr**
**contact@g-echo.fr**
Toulouse - Paris