NOUVELLE FORMULE | NOUVELLE FORMULE | NOU

programmez.com

Programmez!

Le magazine des développeurs

214 JANVIER 2018

Réussir sa migration vers PHP 7

© Orla

Développeur + Agile + DevOps + Infra as Code + Clean Code = une bonne idée ?

Java 9 Utiliser HTTP/2

Les nouveautés de C# 7.1

Les plateformes IoT open source

Créer sa borne d'arcade

C++ Coder son propre IDE partie 2

M 04319 - 214 - F: 6,50 € - RD

LE SEUL MAGAZINE ÉCRIT PAR ET POUR LES DEVELOPPEURS

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

ikoula
HÉBERGEUR CLOUD

PRÉSENTE

CLOUD**IKOULA**ONE

Le succès est votre prochaine destination

MIAMI SINGAPOUR PARIS
AMSTERDAM FRANCFORT

*CLOUD**IKOULA**ONE* est une solution de Cloud public, privé et hybride qui vous permet de déployer **en 1 clic et en moins de 30 secondes** des machines virtuelles à travers le monde sur des infrastructures SSD haute performance.

www.ikoula.com sales@ikoula.com 01 84 01 02 50

ikoula HÉBERGEUR CLOUD

NOM DE DOMAINE | HÉBERGEMENT WEB | SERVEUR VPS | SERVEUR DÉDIÉ | CLOUD PUBLIC | MESSAGERIE | STOCKAGE | CERTIFICATS SSL

Les bonnes rés(v)olutions que l'on oublie dès le 4 janvier*

1. Promis, juré je vais changer mon régime alimentaire pizza – coca –café – fraises Tagada par Dragibus, coca, pizza et café,
2. Je ferai plus de tests… avant d'acheter ma nouvelle console,
3. Je vais faire du code propre tout bien comme il faut (depuis que j'ai lu le dossier code propre inclus dans ce numéro j'ai presque honte de mon code),
4. Oui j'appliquerai le RTFM avant d'installer le moindre matériel (bon ok, là il le faut, j'ai déjà cramé 3 SSD et un écran 4K),
5. J'introduirai des mots compréhensibles pour les non-développeurs au bureau et pour mes amis non-geek,
6. Je ne spoilerai pas, je ne lirai pas les spoilers,
7. Le faire vraiment quand je dirai au chef de projet "ouais je vais le faire",
8. Ne pas rire quand on me dira que HTML est un langage de programmation,
9. Ne pas rire (bis) quand on me dira que Java est aussi rapide que C++,
10. Je changerai de t-shirt chaque jour (#include chaussettes #include slip)
11. Non, tu n'insulteras pas quand on te dira que l'Amiga c'est mieux que l'Atari ST (désolé mais là franchement non, ST forever),
12. Je ne lancerai plus de débat pour savoir si Le réveil de la force n'est qu'un copier-coller de l'épisode 4,
13. Oui tu répareras les PC et avec le sourire (ah la fameuse blague No, I will not fix your computer),
14. Tu seras gentil avec les sysadmins (oui je prends sur moi),
15. Je continuerai à lire chaque mois Programmez!, au réveil, dans le train, au bureau, aux toilettes (oui celle-ci je vais m'y tenir),

Bon code pour l'année 2018.

** on aurait pu dire dès le 1er mais bon, faut laisser un peu d'espoir :-)*

François Tonic
ftonic@programmez.com

SOMMAIRE

Dans le prochain numéro !
Programmez! #215, dès le 2 février 2018

INTELLIGENCE ARTIFICIELLE
Entre peur et réalité

WEB
Utiliser les WebExtensions

Nvidia DIGITS DevBox. Ce monstre est dédié à la simulation, au Deep Learning, à l'IA. Il inclut 4 cartes Titan (12 Go de ram x 4), 64 Go de ram (un peu mesquin pour le coup), Core i-7, seulement 250 Go de SSD. Mais bon, nous sommes dans une station très haut de gamme qui se vend à +11 000 $. Bien plus chère mais à l'équipement hors norme, nous trouvons les DGX Stations, à partir de 70 000 $.

Swift est le langage poussé par Apple mais est-il utilisé en interne ? Il semble que 2017 soit l'année d'une certaine bascule pour les équipes internes. Un développeur s'est amusé à lister les apps codées en Swift sur iOS et macOS. Résultat : 20 apps en Swift dans iOS 11 contre 4 auparavant, 23 apps dans macOS 10.13.x contre 10 un an plus tôt.
Pour voir l'étude : https://blog.timac.org/2017/1115-state-of-swift-ios11-1-macos10-13/

Base lunaire. La lune redevient à la mode. Il était grand temps de s'y intéresser de nouveau. Il s'agit d'une étape importante avant d'aller sur Mars : installer une base permanente, test des matériaux et matériels, formation du personnel. La Nasa est tout naturellement chargée du projet mais reste à définir le budget et l'agenda. Une chose est certaine : les USA ne pourront pas y aller seuls. L'installation des humains dans l'espace est sans doute indispensable pour la survie de l'humanité d'ici 100 à 200 ans.

iOS : les précommandes d'apps sont désormais disponibles pour les développeurs.

2019. Patience, c'est la date probable de diffusion de l'ultime saison de Game of thrones. Pleurez !

La saison 3 de **The man in the high castle** arrivera début 2018. Et la série Electric Dreams, de K. Dick arrivera sur Amazon dès le 12 janvier ! Nous sommes impatients. Et ne râtez pas la fin de la saison 1 de Star Trek Discovery, toujours en janvier. La saison 2 est d'ores et déjà prévue !

Stéphane Richard, patron d'Orange, s'est prononcé contre la neutralité du Net et a même évoqué cela comme une obligation et veut éviter toute considération politique (pourtant les politiques doivent considérer ce problème et agir en faveur de la neutralité). Depuis quelques mois, la neutralité du web refait débat notamment aux USA. Et si le principe d'égalité avait été introduit par la FCC en 2015, aujourd'hui, ce principe est suspendu pour permettre de différentier les flux, les vitesses des réseaux et d'instaurer finalement un accès aux réseaux à plusieurs vitesses. Si pour le moment la neutralité du Net résiste, cette nouvelle offensive est sérieuse et particulièrement forte. Des partisans prestigieux ont signé une lettre pour exiger de conserver la neutralité du web dont Berners-Lee, Cerf ou encore Worniak.
Soyons réaliste : la neutralité du Net sautera à un moment donné. Si des pays technologiques l'actent, d'autres suivront.

Le futur de Java. Le projet OpenJDK continue à travailler sur les 2 prochaines versions : 10 et 11. Pour le moment, la v10 est attendue pour mars 2018 avec un certain nombre d'évolutions et de rajouts déjà prévus. Par contre pour la v11, aucune information très précise n'est disponible.

Java EE est mort, vive EE4J. Après des mois de flou, Oracle a choisi de confier Java EE, la version entreprise de Java, à la fondation Eclipse. C'est un changement majeur qui va s'opérer dans le monde Java, avec sans doute de profonds bouleversements. La fondation Eclipse renforce ainsi sa position dans le monde Java. Les défis à relever par Java et surtout Java EE sont nombreux : le rendre plus flexible, l'alléger, l'adapter aux nouvelles architectures.
Les changements n'ont pas tardé avec une nouvelle marque : EE4J (Eclipse Enterprise for Java). Un des arguments est le fait qu'il s'agisse d'un nouveau projet, d'une nouvelle gouvernance. Java EE n'est plus directement gouverné par le monde Java, il fallait donner un signal fort. EE4J sera compatible avec Java EE, selon la fondation Eclipse. Tous les détails ne sont pas réglés notamment sur la licence qui sera utilisée. Les projets Java EE seront migrés sur EE4J, notamment Glassfish. Oracle continuera à supporter la plateforme mais son rôle actif sera bien moindre.
Reste à préciser l'avenir de Java…

La **Nasa** rallume les moteurs de la sonde Voyager 1. Cet exploit a été réalisé alors que les moteurs n'avaient pas été démarrés depuis 37 ans et que la sonde se situe à +21 milliards de km de la terre ! L'objectif était de tester les petits propulseurs qui permettent de réaligner l'appareil. Cela permettra de repositionner les antennes de transmissions. Même à la vitesse de la lumière, le signal entre le centre de la Nasa et Voyager met presque 20 heures ! Et encore, 21 milliards de KM ce n'est qu'un minuscule grain de sable dans l'immensité de l'univers…

JANVIER

Azure Red Shirt Dev Tour : 23 janvier - Paris

Le nouveau salon

Microsoft organise à l'intention des développeurs un évènement gratuit dénommé Azure Red Shirt Dev Tour et dont l'objectif principal est de permettre à ces derniers de résoudre les problématiques de développement auxquelles ils font face dans le cadre de leurs activités professionnelles. L'évènement, qui s'adresse à tous les développeurs quel que soit leur niveau, sera animé par Scott Guthrie.
A partir de 9h30.

FÉVRIER

Tech Inn Vitré : du 2 au 4 février à Vitré

Ce nouveau rendez-vous est proposé aux **professionnels et au grand public** curieux de connaître **les dernières innovations numériques.** La première édition aura pour thème **la robotique et l'intelligence artificielle.** Cet évènement proposera un programme de **conférences, un hackathon, des ateliers, des stands de démonstration et des animations.** Un programme de conférences et tables rondes sera présenté le vendredi et le samedi pour permettre à chacun de **s'informer sur l'évolution des technologies et de leurs usages**. Chercheurs, roboticiens, industriels et usagers partageront leurs connaissances et leurs expériences afin d'apporter une réflexion collaborative sur **les enjeux de la robotique et de l'intelligence artificielle.** Des ateliers (programmation, robotique, électronique) seront proposés.
Lieu : Centre Culturel Jacques Duhamel 6 rue de Verdun, 35500 Vitré

COMMUNAUTÉS

ParisJUG annonce :
09/01/18 - YoungBlood V

AVRIL

Add Fab : 11 & 12 avril – Paris

Pour sa deuxième édition, Add Fab rassemblera les 11 et 12 avril 2018, Porte de Versailles, **les acteurs les plus importants de l'industrie de l'impression 3D ou Fabrication Additive**. Regroupant les sociétés les plus représentatives et novatrices du secteur, de la startup à l'entreprise internationale, afin de présenter une offre exhaustive du secteur : logiciels, imprimantes, prototypage, matériaux, équipements, outillage et formation. En parallèle se déroulera un cycle de **conférences**, d'**ateliers** et de **formations** en direct.

MakemeFest Angers revient en avril !

Le nouveau salon maker reviendra en avril à Angers. L'évènement 2017 avait réuni +70 startups et makers, + 3000 visiteurs. Pourquoi pas vous ?
Site officiel : http://makeme.fr

Rejoignez
notre cordée !

elcimaï / LE GROUPE

INFORMATIQUE

LA PERFORMANCE
NE DOIT RIEN AU HASARD

Melun

Paris

Elcimaï, éditeur informatique en plein développement
recrute de forts potentiels sur Melun et Paris

Vous êtes :

- **Ingénieur études et développement .Net C#** H/F
- **Ingénieur support applicatif** H/F
- **Ingénieur d'études C / C++ Unix** H/F
- **Ingénieur études et développement Java** H/F
- **Ingénieur études et développement Sharepoint** H/F
- **Consultant AMOA banque** H/F
- **Consultant AMOA assurance** H/F

Retrouvez toutes nos opportunités de carrières sur
www.elcimai.com - E-mail : **candidature@elcimai.com**

Vous interviendrez sur :
Expertise Technique (MOE) C / C++ ;
C#, .Net, JAVA, J2EE, sharepoint…
Expertise et assistance Métier/ Fonctionnelle
(MOA et AMOA) : banque et assurance

François Tonic

ATtiny : un microcontrôleur basique, mais efficace !

Dans le monde Maker, nous avons l'habitude d'utiliser des cartes Arduino, des cartes de prototypages de toutes sortes. C'est pratique et rapide à installer. Mais pour de petits objets ou des projets légers, a-t-on réellement besoin d'une carte entière parfois encombrante, même au format Nano ?

L'idée m'est venue en testant quelques concepts pour optimiser le projet de la tasse connectée présentée en 2016. Finalement, en cherchant un peu, je me dis qu'il est possible de se passer d'une carte Arduino entière pour directement passer au microcontrôleur. Pour pénétrer dans ce monde, j'opte pour une version particulièrement légère : l'ATtiny85.

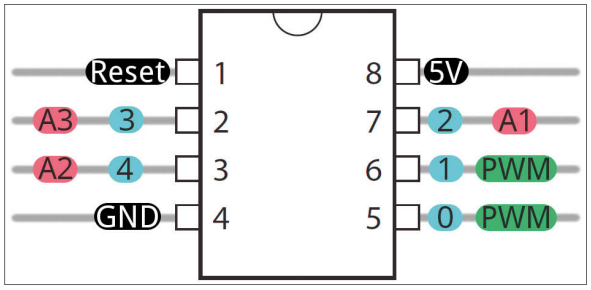
Un microcontrôleur réduit au strict minimum !

Cette puce n'est pas dédiée aux projets ayant de nombreux capteurs, mais pour jouer directement avec le microcontrôleur et commencer à optimiser le PCB et les montages, il s'agit d'une option intéressante. L'Attiny85 a l'avantage d'être très léger et économe dans sa consommation (entre 2,5 à 3,6 mA en utilisation normale et 0,2 à 0,5 µA en sommeil). Son côté minimaliste se voit immédiatement dans les spécifications :

- 8 Ko pour les programmes,
- Vitesse 8 MHz,
- 512 octets SRAM & EEPROM,
- Support SPI et I2C,
- 8 broches.

Sur les broches, on dispose des broches VCC et GND. Vous pouvez utiliser les broches en analogique et en digital. Et il est compatible avec l'Arduino IDE.

Attention : il existe plusieurs modèles d'ATtiny. Par exemple, le modèle 84 propose 14 broches.



Quelques euros par lot de 5 !

Ces microcontrôleurs sont facilement trouvables pour quelques euros un peu partout sur internet. Ils sont livrés avec ou sans socket. Pour faciliter les manipulations, je conseille d'utiliser un extracteur de composants. Cela évitera de tordre les broches, relativement fragiles.

En général, ils sont livrés vierges, sans le moindre bootloader. Le bootleader est ce qui permet de faire fonctionner le code sur la puce. Sans lui, rien ne fonctionne.

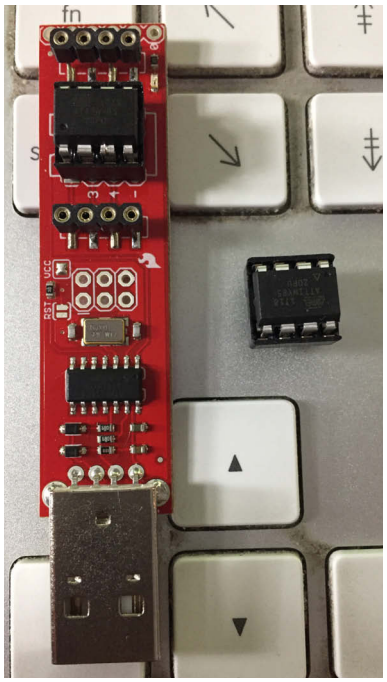
1 Charger le bootloader

La première étape incontournable est de « graver » le bootloader sur le microcontrôleur. Deux possibilités :

- utiliser une Arduino en mode AVR ISP (In-System Programmer) avec une planche à pain,
- un Tiny AVR Programmer.

Personnellement, j'ai opté pour le Tiny AVR Programmer de Sparkfun. Il n'est pas obligatoire. Vous pouvez passer directement via une Arduino. Cette solution est plus économique si vous disposez déjà des composants nécessaires. Nous avons rencontré beaucoup de problèmes de stabilité avec cette solution. L'avantage de passer par un programmer est d'être dans un environnement prêt-à-l'emploi. Il suffit de mettre le microcontrôleur sur le socket (en respectant le sens). L'autre avantage est que l'on dispose de toutes les broches d'ATtiny sur la carte. Idéal pour démarrer un projet. Il faut tout d'abord installer le dossier attiny pour pouvoir reconnaître le microcontrôleur par l'IDE : https://github.com/damellis/attiny. Le dossier attiny devra être placé dans le dossier Arduino. Lancez l'IDE.

Si tout va bien, dans le menu board, les attiny sont visibles. On sélectionne ATtiny85 (1 ou 8 MHz). Le port sera celui reconnu par l'outil et dans la partie programmer : USBtinyISP. Puis on lance la « gravure » du bootloader. L'opération prend quelques secondes. Voilà, l'ATtiny est prêt.



2 Et ensuite ?

La suite c'est à vous de l'imaginer. Pour charger les codes, c'est très simple : on téléverse et c'est tout. Par contre attention à ne pas dépasser le voltage du microcontrôleur, vous risquez de le griller. Et n'oubliez pas que vous êtes limité dans le nombre de broches. Par contre, vous pouvez parfaitement installer la puce dans des maquettes, de petits objets avec 1 ou 2 capteurs. A vous d'imaginer le support sur lequel vous allez installer l'ATtiny ou même le PCB. Vous aurez une grande liberté. En revanche, si vous devez recharger du code, mettez la puce sur un socket ce sera plus simple. Pour l'alimentation, on utilisera les broches VCC et GND. Le plus courant est d'installer une pile bouton (via un mini shield). Le principe est le même pour les ATmega128 ou 328. Dans ce cas, vous aurez bien plus de souplesse sur la connectivité comme on retrouve le principe d'une Uno ou d'une Nano, mais attention le montage pour flasher le bootloard sera un peu plus compliqué. Pour aller plus vite, vous pourrez passer par un shield programmer. A vous de jouer ! •

VOUS DÉVELOPPEZ AVEC WINDEV. PENSEZ À AUDITER VOS APPLICATIONS !

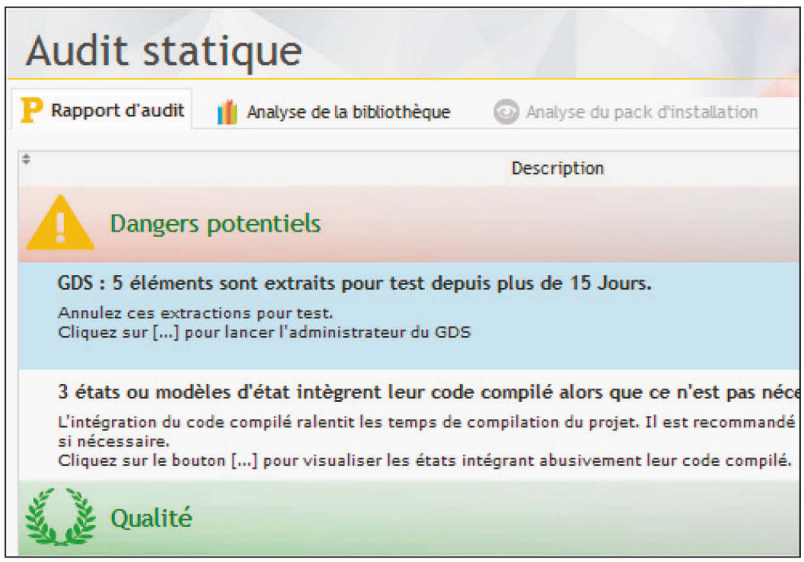
Les fonctionnalités d'Audit automatique présentes en standard dans WINDEV vous permettent d'accroitre la fiabilité des applications que vous créez, et d'en accélérer la vitesse. Prenez quelques minutes pour auditer vos applications, vos utilisateurs vous remercieront ! Il existe 2 types d'Audits: l'Audit , pour vos sources, et l'Audit Dynamique pour les applications déployées

AUDITEZ VOS SOURCES D'UN CLIC

L'audit statique analyse le projet source et les éléments qui le composent. Vous réalisez l'audit de vos applications aussi souvent que vous le désirez.

Le rapport détaillé qui est édité indique: contenu de l'install, métriques de code, dangers potentiels pour le projet, éléments non utilisés, conseils d'optimisation...

Vous pouvez ainsi **améliorer la qualité de vos applications**, réduire leur taille, **augmenter la sécurité**...



Un exemple de rapport d'audit statique: ici la partie «dangers potentiels du projet», à faire corriger rapidement !

DES APPLICATIONS PLUS RAPIDES ET PLUS ROBUSTES

NETTOYAGE DE PROJET

Après nettoyage, vos projets sont plus petits et plus rapides.

Les éléments suivants sont proposés au nettoyage:

- Code mort
- Fenêtres inutilisées
- Variables inutilisées
- Messages multilangues inutilisés
- Sauvegardes simultanées de l'historique dans l'éditeur et dans le GDS
- Styles non utilisés
- Groupes de champs non utilisés.

MÉTRIQUE DE CODE

- Taux de commentaires
- Utilisation d'asserts
- Nombre de lignes par traitement (détection des excès)
- ...

DANGERS POTENTIELS POUR LA QUALITÉ DU PROJET

L'audit statique génère une liste de dangers potentiels sur le projet. Parmi les sujets relevés, on trouve:

- Eléments du projet extraits depuis une longue période et jamais réintégrés
- Modèle (de champs, de fenêtres, de pages, d'états,...) pas à jour
- Gabarit utilisé par le projet absent sur la machine
- Connecteur Natif à une base tierce utilisé par le projet mais non présent
- Configuration de projet contenant plusieurs générations
- ...

AUDITEZ L'EXÉCUTION RÉELLE

L'audit dynamique est effectué sur une application en exécution. L'exécution a lieu sur une machine de tests ou sur une machine d'exploitation.

L'audit dynamique permet de suivre l'exécution d'une application sur un poste, ainsi que l'occupation mémoire.

L'audit génère un rapport contenant warnings d'exécution, asserts, dumps mémoire, ...

Couplé au profiler, l'audit dynamique vous permet d'**augmenter la vitesse de vos applications**.

AUDIT DYNAMIQUE (APPLICATION EN EXÉCUTION)

Un audit dynamique est effectué sur une application en exécution, sur une machine d'exploitation ou sur une machine de test.

Il n'est pas nécessaire d'être présent physiquement sur le site d'exploitation où l'audit est effectué.

L'audit génère un rapport listant les dangers et les optimisations potentielles, ...

La génération du rapport peut être demandée par programmation, ou directement par l'utilisateur (combinaison de touches [Ctrl] [Alt] [A]).

Ce fichier peut être analysé à distance.

OCCUPATION MÉMOIRE

En exécution d'application, en local ou à distance, analysez l'occupation mémoire d'un ordinateur.

WARNINGS D'EXÉCUTION

Voici une liste (non exhaustive) d'éléments qui sont analysés et rapportés par un audit dynamique:

- Images non trouvées
- Fichier dont le chemin d'accès n'existe plus (time-out)
- Cas non existant dans un SELON
- Dépassement de capacité des opérations
- Valeurs tronquées dans les affichages (et qui produisent donc des affichages de "++++")

...

ERREURS NON FATALES

L'audit dynamique détecte et liste ces erreurs «invisibles».

François Tonic

La réalité virtuelle au cœur de la pyramide de Chéops

Depuis 2 ans, une vaste mission scientifique utilise les technologies les plus modernes pour explorer et répondre à quelques-uns des mystères de la grande pyramide de Gizeh, près du Caire. Cette pyramide remonte au règne du pharaon Chéops, vers 2 550 av. J-C. Ce monstre de pierre faisait à l'origine 146 mètres de haut. Les anciens égyptiens ont su orienter le monument sur les points cardinaux avec une rigueur étonnante et les inclinaisons des faces sont parfaites. Surtout, c'est la seule pyramide à posséder un aménagement intérieur aussi complexe et notamment la monumentale Grande Galerie au coeur de la construction : 47 mètres de longueur, 8,6 de hauteur, 26° d'inclinaison… Le monument a suscité d'innombrables théories et personne ne sait si des salles inconnues restent à découvrir. Scan Pyramids veut le savoir !

Scan Pyramids utilise plusieurs technologies pour osculter le monument sans le détériorer. Une des techniques les plus spectaculaires est la détection des muons, des particules cosmiques qui bombardent la Terre constamment. Des détecteurs et téléscopes à muons ont été déployés durant plusieurs mois dans et autour de la pyramide. Une grande masse de données a été récoltée. Mais comment les présenter au public ? Comment les modéliser et rendre accessibles les résultats ?

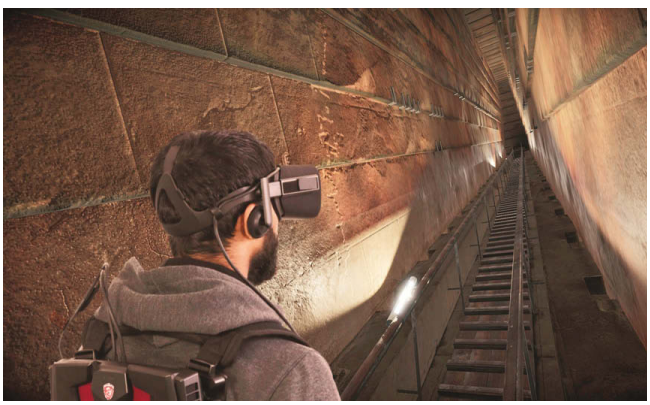
Emissive, société parisienne dans la réalité virtuelle et la 3D, s'est rapidement intéressée à cette mission. La société connaît depuis 10 ans la pyramide de Chéops car elle a participé à un autre projet, Khéops révélé. Emissive a mobilisé une personne à plein temps et le travail a été considérable : concevoir des modèles numériques et virtuels de la pyramide à partir des plans les plus précis connus, faire des relevés sur place, relever un maximum d'informations comme la texture des pierres. Un des défi a été de recréer un modèle 3D du plateau de Gizeh, qui n'existait pas, puis s'occuper de la pyramide elle-même.

Ces modèles permettaient de visualiser et de "manipuler" les données récoltées par la mission. Car les muons capturés ressemblent à un amas de points qu'il faut être capable de lire et de visualiser.

La réalité virtuelle au service de l'archéologie

L'utilisation de la VR n'était pas réfléchie dès le départ mais elle est venue au fur et à mesure des mois avec les données et informations récoltées (muons, prises de vues, photogramétrie, etc.). Emissive avait déjà réalisé plusieurs projets pour des magasins et la technologie étant de plus en plus mature, la VR s'est peu à peu imposée comme interface de navigation et d'immersion. Et le grand public connaît désormais cette technologie. L'autre objectif était d'être en immersion dans la pyramide et pas uniquement de faire une visite en 360° qui n'aurait pas apporté grand chose.

Le projet était dès le départ ambitieux car il s'agissait de recréer la pyramide, de créer un scénario de visite, avoir plusieurs personnes simultanément dans la "visite", le tout dans un espace de 400 m2



pour ressentir l'immensité du monument mais aussi "subir" le monument comme les couloirs très bas de plafond, obligeant à marcher accroupi. L'autre défi était de proposer une expérience totalement autonome. Des sacs dédiés à la VR contenant la puissance de calcul et les batteries ont été utilisés pour faciliter les déplacements des personnes.

Pour réduire les coûts et couvrir une grande surface, Emissive a choisi le système Optitrack, bien connu dans la motion capture. Un Vive ne dépasse pas une surface 7-8 m2. Pour gérer les mouvements, les personnes, leurs positions dans l'espace, les caméras bordant la salle repèrent les casques VR. Ces casques, des Oculus, sont équipés d'un cover composé de leds actives reconnues par la caméra. Le premier prototype a été mis au point en juin dernier mais la phase bêta a pu démarrer véritablement en automne. Ce système permet une autonomie d'environ 1h ce qui est largement suffisant pour effectuer la visite.

Unity pour la partie développement

Si les développeurs connaissent Unreal Engine et le C++, il a été décidé de passer entièrement à Unity. Le développement est fait en C# et un système SVN est utilisé pour gérer les sources. Un socle technique commun a été mis en place comme cela tout le monde utilise les mêmes briques techniques et la même version d'Unity (évitant d'éventuels problèmes de versions et de builds). Visual Studio et Mono Develop sont les deux IDE utilisés.

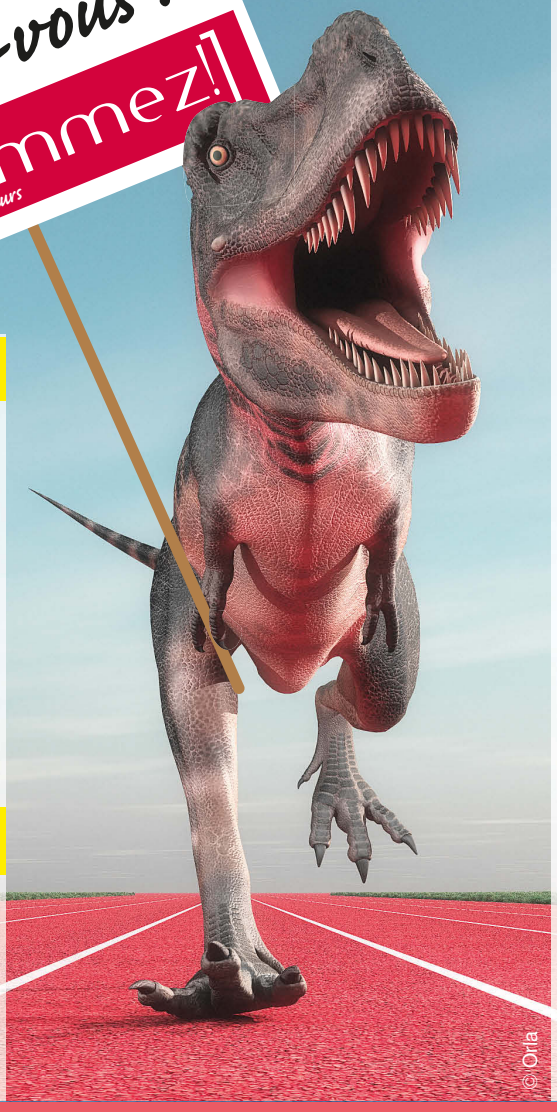
Un des problèmes initiaux fut le scénario : proposer une visite "intelligente" de la pyramide et des résultats de la mission Scan Pyramid. Une fois résolue, l'autre difficulté a été de gérer plusieurs personnes en même temps dans le système : latence, réseau, connexion / déconnexion, monitoring de chaque sac, etc. Aujourd'hui une vingtaine de personnes peuvent participer en même temps à cette expérience !

Début novembre, le protocole grandeur réelle a été montré à la Cité de l'Architecture de Paris. L'objectif est de proposer une animation pour 30 personnes courant 2018 quand tous les petits soucis techniques seront réglés et que le financement suivra.

Pour aller plus loin : Pharaon Magazine n°32 à paraître le 20 janvier 2018

Merci à l'équipe d'Emissive.

•



Nos classiques

1 an ..... 49€*
11 numéros

2 ans ..... 79€*
22 numéros

Etudiant ..... 39€*
1 an - 11 numéros

** Tarifs France métropolitaine*

Abonnement numérique

PDF ..... 35€
1 an - 11 numéros

Souscription uniquement sur
www.programmez.com

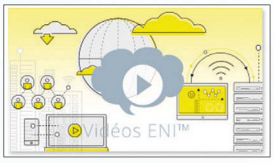
Option : accès aux archives 10€

Nos offres d'abonnements 2018

1 an ..... 59€
11 numéros
+ 1 vidéo ENI au choix :

- Arduino*
  Apprenez à programmer votre microcontrôleur
- jQuery*
  Maîtrisez les concepts de base

eni Editions



2 ans ..... 89€
22 numéros
+ 1 vidéo ENI au choix :

- Arduino*
  Apprenez à programmer votre microcontrôleur
- jQuery*
  Maîtrisez les concepts de base

Offre limitée à la France métropolitaine
** Valeur de la vidéo : 34,99 €*

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

☐ **Abonnement 1 an :** 59 €
11 numéros + 1 vidéo ENI au choix :
☐ **Abonnement 2 ans :** 89 €
22 numéros + 1 vidéo ENI au choix :

☐ Vidéo : Arduino
☐ Vidéo : jQuery

PROG 214
Offre limitée, valable jusqu'au 2 février 2018

☐ Mme ☐ M. Entreprise : Fonction :

Prénom : Nom :

Adresse :

Code postal : Ville :

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

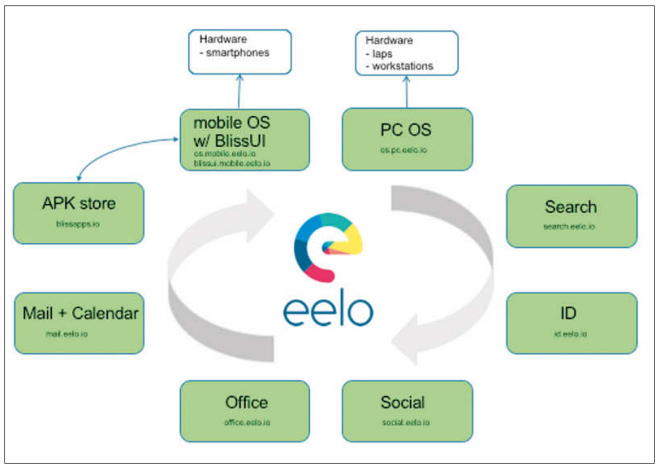
E-mail : @

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

*** Tarifs France métropolitaine**

Gaël Duval *est un des acteurs de l'open source français depuis 20 ans. En 1998, il crée la base de la distribution Mandrake qui a fait le bonheur de millions d'utilisateurs. La distribution avait su se faire un nom auprès des Suse, Fedora, Ubuntu. MandrakeSoft est créé pour poursuivre le développement. Peu à peu, Gaël s'éloigne de la société et lance en 2007 une nouvelle société : Ulteo. Puis en 2016, il se tourne vers un nouveau projet Nfactory, avant de lancer eelo.*

eelo : une plateforme mobile 100 % open source



Quelle était ton idée de départ du projet eelo ? Ne plus dépendre d'un Google ou d'un Apple ?

J'ai réalisé il y a quelques temps que j'utilisais maintenant un Mac depuis deux ans, un iPhone depuis dix ans, Google de plus en plus, Facebook... Et que toutes ces boites passent leur temps à aspirer mes données personnelles et à en tirer un profit commercial indirect mais massif. Et je constate que c'est la même chose pour mon entourage, mes amis, ma famille, et plus largement ma communauté d'intérêts (la France, l'UE). Pour résumer cette situation j'aime reprendre la phrase de Pierre Bellanger : "les réseaux sociaux sont en Californie, les plans sociaux en Picardie".

Pour moi c'est difficilement acceptable d'être arrivé, par fainéantise, à cette situation de servitude passive.

Donc je me suis dit que j'allais voir ce qu'on pouvait faire, au minimum pour moi, pour créer un environnement numérique agréable et homogène qui n'enverrait plus (ou le moins possible) mes données personnelles nourrir l'économie américaine.

C'est de cette démarche, peut être un peu utopique, qu'est né le projet eelo.io - et je me suis aperçu en commençant à en parler que je ne n'étais pas seul à raisonner comme ça. Pour résumer, le projet d'eelo c'est proposer un système open source respectueux des données personnelles. Attractif, et pour tous.

Tu dis, dans les présentations du projet, que l'on dispose de toutes les briques techniques et technologiques pour créer un système mobile entier ? Quelles sont ces briques et comment tu les utilises ensemble ?

Je vois le projet comme un tout :

- système d'exploitation mobile,
- services Internet associés.

Donc j'ai commencé à m'intéresser à la première brique : le système d'exploitation pour mobile. Il y avait plein de possibilités : repartir sur FirefoxOS, Ubuntu, AOSP, LineageOS... Finalement j'ai retenu LineageOS, un fork d'Android qui est supporté sur pas mal de devices et surtout permet de proposer des applications couramment utilisées, sans devoir passer par Google Play Store. Mais LineageOS ce n'est (à mon goût) pas très joli au niveau interface. Donc il faut faire plus attractif, c'est du développement.

Ensuite je me suis intéressé à ce qu'on pouvait proposer sur les services internet. On a besoin a minima d'un mail, d'une suite office en ligne performante, d'un système de cloud personnel... Je pense aujourd'hui que toutes les briques sont disponibles, même si elles sont disparates. Il faut les homogénéiser et les intégrer pour en faire un truc cohérent ultra-simple à utiliser qui respecte à la fois la vie privée et l'utilisation des données personnelles.

Plusieurs projets mobiles sont apparus depuis 10 ans tels que ceux de Mozilla et d'Ubuntu, les deux furent des échecs. On peut aussi citer Meego ou encore Tizen parmi les échecs. Penses-tu avoir trouvé une bonne approche ?

C'est frappant car je me suis énormément intéressé à FirefoxOS il y a environ trois ans, mais à l'époque mon projet aurait été de créer un SDK applicatif et un store pour cette plateforme. Car je sentais que l'OS était top mais que ça coinçait au niveau des applications, car les utilisateurs veulent avant tout des applications.

Et avant que j'aie pu concrétiser l'idée, FirefoxOS est mort, précisément car il n'y avait pas assez d'applications et que des éditeurs d'applications massivement utilisées dans leur marché cible ont refusé de porter sur FirefoxOS. C'est le même problème pour beaucoup de plateformes.

Et précisément, je me suis décidé à me lancer dans eelo quand j'ai réalisé qu'on pouvait avoir toutes les applications courantes sur LineageOS sans même passer par Google Play store.

Côté agenda, comment le projet va-t-il se lancer, se créer ?

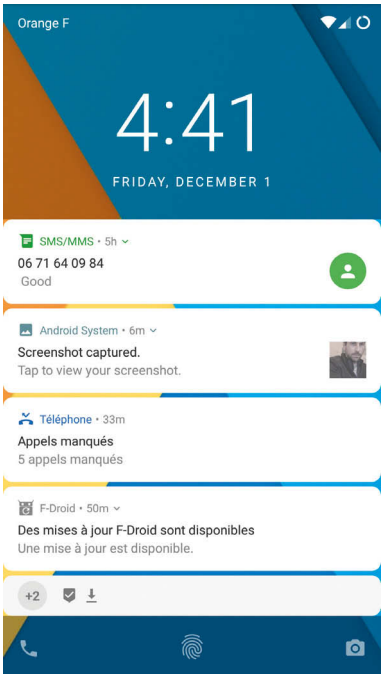
On a déjà un prototype sur le mobile, avec une nouvelle interface graphique : nouveau launcher, un peu plus à la "iOS", nouveau lockscreen, nouvelles notifications... Premier objectif dans les 6 prochains mois : arriver à un MVP avec OS mobile qui tourne sur un certain nombre de devices + les web services associés. Si on arrive à ça on aura prouvé que c'est possible et ce sera

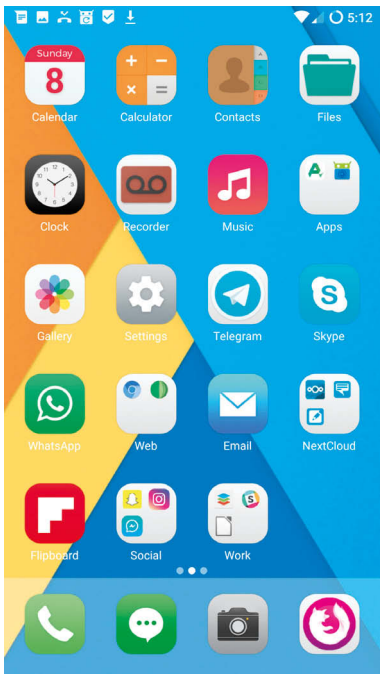
Modèle de développement

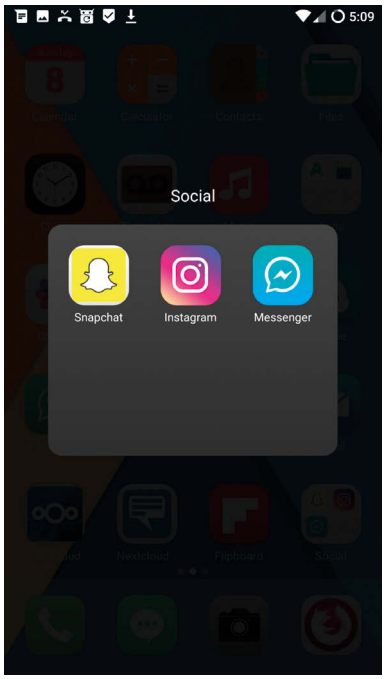
Le système étant basé sur un socle Android, le modèle de programmation est celui d'Android : Java. Mais le projet se pose la question de créer un SDK indépendant. Si SDK il y a, il sera pas disponible immédiatement.

Quelques éléments :

https://www.indidea.org/gael/blog/leaving-apple-google-eelo-odyssey-part1-mobile-os/
https://www.indidea.org/gael/blog/leaving-apple-google-eelo-odyssey-part2-web-services/#more-1066







plus simple pour mobiliser une communauté, déjà naissante d'ailleurs.
Et fin 2018 ce seront les premiers smartphones pré-chargés avec eelo. Plutôt du haut de gamme pour commencer, mais l'objectif c'est aussi à terme de proposer un système qui permette de faire vivre tous les devices pas forcément récents. C'est l'agenda de la campagne kickstarter en cours.

Pour les développeurs, qu'est-ce qui sera proposé concrètement ? Car sans développeur, difficile de réussir non ?
C'est un projet open source et sans but lucratif. Donc concrètement un contributeur a en retour ce qui est produit par le projet. Mais on va quand même financer quelques développeurs (et pas seulement) pour aller vite, surtout au début, c'est le sens de la campagne de financement participatif.

Côté utilisateur, ce système sera-t-il installable facilement ou des smartphones eelo seront-ils proposés ?
On va proposer des ROMs installables, plutôt par des connaisseurs, et des smartphones pré-installés, avec mises à jour automatiques. C'est loin d'être trivial de flasher un smartphone aftermarket, c'est plus compliqué que d'installer une app ! •

LA PARTIE ARCHITECTURE

L'OS mobile retenu est donc LineageOS, qui a besoin d'être adapté sur plusieurs aspects. L'interface graphique tout d'abord, a été en partie modifiée pour davantage ressembler à ce qu'on trouve sur iOS ou MIUI. En particulier, un nouveau launcher a été créé "BlissLauncher", qui remplace par défaut "Launcher3" (Trebuchet) de LineageOS. Il est intégré directement dans les sources de la nouvelle ROM eelo. Un nouveau jeu d'icones est également proposé. En parallèle, un nouveau "Lockscreen" a été développé, qui supporte les notifications standard, et est intégré également par défaut dans les sources avant recompilation afin de pouvoir accéder aux appels bas-niveau avec des droits suffisants. Le centre des réglages ("settings") va être repensé à terme pour être plus simple à utiliser dans son organisation. Dans un premier temps ce sera d'abord un lifting visuel.
Une spécificité d'eelo étant de ne pas utiliser de services Google, on n'installe ni application Google, ni même les Google Play Services. Etant donné que ces derniers sont nécessaires pour un certain nombre d'applications pour fonctionner, on les remplace par MicroG qui propose la même ABI. Et afin de contourner SafetyNet de Google (qui permet aux applications de vérifier qu'elles sont dans un environnement "conforme", c'est à dire pas dans une ROM eelo ou autre...), on intègre Magisk. Et pas mal de modifications sont faites directement au niveau des sources ou des settings par défaut afin de :

- proposer des résolveurs DNS par défaut plus respectueux de la vie privée (9.9.9.9) ;
- proposer des moteurs de recherche ou des méta-moteurs de recherche différents (DuckDuckGo, Qwant et peut-être un jour CommonSearch:) ;
- proposer une mise à jour automatique eelo ;
- proposer un store d'applicatifs eelo (qui combine les APKs libres de F-droid et les APKs non libres mais gratuits et sûrs de APKPure) ;
- proposer une sauvegarde en ligne sécurisée pour les photos, les notes, les réglages utilisateurs. Par défaut on va proposer un cloud eelo utilisant NextCloud, et la possibilité pour l'utilisateur de régler tout ça sur un autre cloud provider, éventuellement auto-hébergé.

La question se pose évidemment pour eelo de devenir ID-provider, afin que les utilisateurs puissent se créer un compte unique qui pourrait être utilisé pour tous les services : le cloud perso, online-office, le mail etc. On étudie les meilleures solutions pour faire ça.
Se pose également de ce qu'on peut faire pour renforcer la sécurité au niveau du kernel sans nuire à la facilité d'utilisation et aux fonctionnalités. C'est un vrai challenge.
Un aspect technique "zone grise" du projet eelo est que sur la plupart des smartphones, on n'a pas d'autre choix que d'utiliser les drivers matériel bas-niveau des constructeurs qui sont tous propriétaires. Donc il va falloir auditer leur comportement. Une alternative sera de proposer également eelo sur FairPhone ou des smartphones 100% open source au niveau du hardware, mais ils sont peu nombreux.
Pour l'anecdote, la première compilation d'une ROM custom prend actuellement environ 3 heures sur un octo-core i7 à 4Ghz doté de 32 GB de RAM. Une recompilation partielle prend ensuite moins de 10 minutes.

• Erik Lenoir
Responsable du pôle Sécurité chez Zenika, Erik est un véritable passionné d'informatique. Son leimotiv est de remettre la sécurité au centre des projets. Profil Javaiste, il apprécie également les technologies back ainsi que les problématiques de performance et d'optimisation.

Plongeons dans les profondeurs de l'Internet : entre Darknets et Cybercriminalité

Benjamin Brown est un chercheur travaillant sur le darkweb, l'évaluation des risques, la gestion des incidents ainsi que la résilience des systèmes chez Akamai Technologies. Il a travaillé pour de grandes multinationales, des académies ainsi que des projets à but non lucratif, il a également un diplôme en étude anthropologique et en sciences internationales.

Ses recherches portent sur des **études ethnographiques du darkweb et du deepweb**, les différents vecteurs d'attaque, les systèmes radio, la psychologie et l'anthropologie dans la sécurité informatique, les techniques métacognitives de l'analyse informationnelle, le profilage des auteurs de menaces, ainsi que sur le fait d'envisager la sécurité dans un système écologique complexe. Le premier **marché noir sur Tor a ouvert il y a 6 ans**. Depuis, cette période a été marquée par une croissance exponentielle ainsi que des innovations et des adaptations. Avec plus de 30 marchés noirs actifs sur le darkweb ainsi que d'innombrables boutiques personnelles, nous sommes très loin d'en voir le bout. Durant sa conférence à la Nuit du Hack 2017, l'auteur explique les **différences fondamentales entre le deepweb, le darkweb ainsi que le darknet**.

Ensuite, Il explore différents frameworks utilisés sur le darknet (c'est vrai, Tor ne sert pas qu'à alimenter la cybercriminalité). Il présente également un peu l'historique en matière de cybercrime, de vente d'herbe sur ARPANET et de groupes de discussions. Nous verrons ainsi les débuts du commerce parallèle sur le darkweb.

Puis l'état actuel du darkweb ainsi que les moyens mis en place par les gouvernements et les forces de l'ordre pour répondre à ce nouveau front du crime. Enfin, il met en avant quelques récentes législations et contrôles qui ciblent la base des technologies utilisées par le darkweb et ses utilisateurs. Pour finir, nous verrons quelques technologies émergentes qui intègreront les prochains systèmes économiques du darkweb. **Quelques définitions importantes pour éviter les amalgames** :

- Le Deepweb est la partie immergée de l'iceberg (qu'est le World Wide Web). Il s'agit de tout ce qui est accessible depuis un navigateur mais qui n'est pas référencé par les moteurs de recherches. Par exemple, lorsque vous accédez à un service privé protégé par un mot de passe (votre compte bancaire en ligne) vous êtes dans le web profond ;
- Les Darknets sont les réseaux restreints et incluent web, email, chat, partage de fichiers et routage obscurci de l'Internet ;
- Le Darkweb représente tous les services accessibles depuis un navigateur web spécifique (par exemple Tor browser).

Ainsi le début des années 70 marque les premières fraudes sur Arpanet avec la vente de marijuana et les vols d'argent via systèmes électroniques dès 1964 (fraude découverte « quelques » années plus tard, en 1973).

Le terme de Darknet avait un sens différent d'aujourd'hui. Il s'agissait des réseaux qui n'envoyaient aucune information à l'Arpanet. Ils pouvaient en recevoir, mais comme Arpanet était militaire, ils utilisaient des pare-feux ou autres pour des raisons de sécurité. Il n'y avait aucun moteur de recherche pour ces réseaux. Pour la petite anecdote, tous ces réseaux périphériques qu'étaient les darknets, sont devenus les pierres angulaires de notre Internet moderne. Les décennies 1980/1990 ont été marquées par l'essor des collectifs de hackers (Chaos Computer Club, Legion of Doom, Chinese Hokners) et des grosses attaques sur les banques et les médias. Ces groupes ont commencé à être reconnus dans les médias, le nombre d'attaques augmentant sérieusement et notamment sur les sites gouvernementaux.

La riposte gouvernementale

Les gouvernements se sont donc trouvés dans l'obligation de répondre à ces vagues de cybercriminalité. En 1981 apparaît le premier séminaire de formation d'Interpol pour les enquêtes sur les cybercrimes.

En 1982, les Etats-Unis d'Amérique votent le terrible « Computer Fraud and Abuse Act » qui est toujours grandement utilisé.

En 1988, les Lois Godfrain arrivent en France ainsi que la formation de l'agence de fraude informatique, aussi connue sous le nom de DST. **La France est la pionnière** dans les pays européens en ce qui concerne les lois et équipes luttant contre la cyber fraude.

En 1989, le Conseil Européen émet des recommandations sur les lois à adopter contre le cybercrime et comment lutter. C'est là-dessus que la plupart du reste de l'Europe va baser ses lois. En 1990, le Royaume-Uni sort le « Computer Misuse Act » et en 1998 le G8 met en place une hotline d'experts enquêteurs pour aider les pays et polices.

Vient alors **l'âge des premières guerres de cryptographie**.

Entre 1990 et 1992 survient la grande répression des hackers (« the great hacker crackdown »), une des plus grandes opérations fut l'opération « Sundevil » avec le FBI, les services secrets et d'autres forces de l'ordre qui se sont réunis pour cibler et mettre à mal les systèmes de bulletins électroniques illégaux, mais également Phrack (qui est un magazine électronique underground toujours actif). La NSA était également de la partie, de nombreux BBS ont commencé à fermer pour éviter les poursuites. Entre 1994 et 1995, les débits des fournisseurs d'accès à Internet augmentent fortement, ce qui ouvre l'apparition des images boards, soft board, et services de messagerie qui viennent tuer les BBS.

Les darknets

Un darknet doit respecter les contraintes suivantes :

- Être un réseau isolé ;
- Être difficilement indexable.

Les **Usenet** sont des systèmes de messagerie où l'on utilise des flux de messages pour communiquer avec les autres. Ils sont tou-

jours utilisables aujourd'hui mais utilisés principalement pour la piraterie. C'est la naissance du spam en masse, de la « sporgery » (qui signifie « forge spam » où l'on usurpe l'identité d'un tiers), les botnets de grande taille (qui sont des spams bots automatiques). Sur le même modèle, les « cancel bots » apparaissent, cherchant à annuler les effets des spams bots, puis les « forge cancels » viennent inhiber les « forge spams ». En 1994, on assiste à une forte diminution de l'usage des Usenet provoquée par 3 raisons principales :

- Apparition du fournisseur AOL, permettant à de nombreux utilisateurs d'accéder à ces Usenet, ce qui provoque une grosse augmentation de personnes qui ne savent pas s'en servir et qui dérangent ;
- Augmentation de la saturation du réseau et du stockage avec toutes ces nouvelles personnes, ce qui provoque un désengouement de la part des fournisseurs d'accès à Internet à l'hébergement de Usenet ;
- Le procureur général de New York sévit sur les Usenet sous motif de la pornographie juvénile que l'on pouvait y trouver. Ce qui provoque la fermeture de nombreux Usenet de la part des fournisseurs d'accès pour ne pas enfreindre la légalité.

Les discussions relayées par Internet (« Internet Relay Chat » ou **IRC**) sont des services de messagerie qui étaient utilisés pour parler de commérage, d'hacking, etc. On pouvait usurper (volontairement ou involontairement) le pseudo d'un tiers, on y trouvait beaucoup de logiciels piratés, d'œuvres sous propriété intellectuelle, livres numérisés, pornographie, malware, IRC bots… C'est à ce moment-là que l'achat et la vente de drogues par Internet explose. Il y en avait très peu sur les BBS et Usenet contrairement aux IRC qui étaient riches en drogues et blanchiment d'argent. Il y était coutume de défier les écoutes de la NSA en ajoutant des mots clefs dans une conversation pour déstabiliser l'espionnage des conversations. Le temps des réseaux pair à pair (« Peer to Peer », **P2P**) correspond à l'explosion de fichiers échangés, et donc d'œuvres sous propriété intellectuelle, de virus, malware… Il s'agissait de darknets par la nature des échanges (décentralisés) et de l'absence d'un vrai moteur de recherches sur ces réseaux à l'époque.

Le **Sneakernet** (ou « Tennis-Net », « Armpit-Net », « Floppy-Net' » ou « ShoeNet ») est une méthode de transfert de fichier sans réseau informatique, qui fonctionne par exemple par l'intermédiaire de clés USB ou de disques durs externes. Ce système est **toujours utilisé** dans certains pays où l'on peut trouver des tables pleines de clefs USB avec des musiques ou films « piratés ». Entre 2003 et 2005, des opérations géantes dénommées « firewall » et « AnglerPhish » sont menées à l'international par la CIA, les services secrets, Interpol et les autres forces de l'ordres pour fermer ces services.

Clearnet Markets

- 2000 : The counterfeit library (diploma mill, ID fraud resources) ;
- 2001 : CarderPlanet (credit card fraud, grosse communauté russe) ;
- 2002 : ShadowCrew (vol de données, évolution de TCL, vol d'identité) ;
- 2003-2005 : Major global LE Focus (opérations Firewall et AnglerPhish) => disparition des gros forums et séparation en nouveaux et petits forums ;
- 2004-2006 : CardersMarket (Iceman/Max Butler, hacked and merged other forums) ;
- 2007-Aujourd'hui : Hacked or shut down BY LE / « Whack-a-mole » fraud forums.

Les monnaies digitales

Avec ces marchés numériques, les devises « en or numérique » font leurs apparitions.

Les principales furent : eGold ; Pecunix ; iGolder.

Leur ambition était de mettre en place **une monnaie mondiale convertible en or**, et donc garantie sur un stock d'or. Aujourd'hui elles sont quasiment éteintes, Pecunix s'est déplacée dans un autre marché et les deux autres ont été stoppées. Entre 2006 et 2013 la plus grande e-devise pour le cybercrime fut « Liberty Reserve » qui était surnommée la banque du marché noir (« The Black Market Bank »). Elle était connue pour le blanchiment d'argent à grande échelle.

Cryptocurrencies

Il y a plusieurs crypto-monnaies que l'on peut choisir. Le choix de Benjamin Brown est incontestablement le Coinye, qui est comique selon lui car Kanye West fut en colère de voir l'illustration basée sur sa tête.

Bitcoin

La crypto-monnaie emblématique date de 2009, personne ne connaît son ou ses réels créateurs (seulement le pseudo de « Satoshi Nakamoto ») et il s'agit de la plus large en termes de capitalisation boursière.

C'est la première monnaie des Darknets.

Elle possède un livre de transactions libre et décentralisé, ce qui fait qu'elle n'est pas anonyme.

Litecoin

Il s'agit d'un fork du Bitcoin, avec tous les problèmes d'anonymat qui vont avec. Il s'agit de la 4ème crypto-monnaie en matière de capitalisation.

Ethereum

Ethereum est à la fois une crypto-monnaie, mais également un protocole d'échanges décentralisés pour construire des applications distribuées, résistantes aux hacks et aux tentatives de mise à mal. Elle a **augmenté de 300% en valeur** au cours du mois de Juin 2017.

Monero

Créée en 2014, elle est **axée sur la vie privée** et donc intéresse le Darknet. C'est la 6ème plus importante capitalisation boursière et sûrement la plus rapide augmentation de valeur. Elle cache des métadonnées et autres informations dans sa chaîne de bloc. Toutes ces crypto-monnaies ont de vrais usages mais également un usage criminel. De très nombreux ransomwares demandent de payer la rançon en Bitcoin, Monero ou une autres crypto-monnaie. Des malwares sont spécialisés dans le minage de ces crypto-monnaies à l'insu de leurs victimes. Il y a même des fonctions de minage dans les botnets IoT, pas forcément très judicieux. Elles sont utilisées dans le blanchiment d'argent et dans l'échange de biens et services illégaux.

Modern darknets

Le tout premier n'avait pas de composant web, seulement des services de messagerie, du partage de fichier, etc. Les trois plus grands, toujours actifs, sont : OneSwarm ; Tribler ; RetroShare.

Puis viennent ceux qui supportent les sites web, à commencer par Tor.

Tor

C'est le plus **populaire**, développé initialement par la DARPA entre 1994 et 1997 mais a été publié sous licence publique. Il utilise un « routage en oignon » (qui lui vaut son nom « The Onion Routing ») avec plusieurs couches de chiffrements. C'est de loin le plus vaste Darknet et la plus importante plateforme pour les marchés du Darknet.

Freenet
Sorti en 2000 et concentré sur la liberté d'expression, il est construit de façon à ce que si un nœud du réseau tombe, le service soit toujours accessible grâce à une redondance des données. Il est basé sur du chiffrement, ce qui augmente la sécurité mais a pour inconvénient d'être plus lent et gourmand en ressources.

I2P (« Invisible Internet Project »)
Sorti en 2003, c'est le second plus important. Il utilise un routage en « ail » qui est une extension du routage en « oignon ». Il n'est pas seulement concentré sur l'anonymat de navigation web, mais l'anonymisation de tous les protocoles réseaux.

Gnunet
Sorti en 2001, ce n'est pas un Darknet par défaut, mais ça en devient un dès qu'on l'utilise en mode Ami à Ami (« Friend To Friend »). Il a fortement gagné en popularité avec les révélations de Snowden.

Netsukuku
Sorti en 2006, c'est le plus petit et **encore en développement**. Contrairement à ce que son nom suggère, les créateurs sont italiens. Il possède son propre protocole de routage QSPN, Quantum Shortest Path Netsukuku (algorithme de routage) et son système de DNS distribué et décentralisé ANDNA (« A Netsukuku Domain Name Architecture »). C'est encore en pré-alpha et il n'y a pas encore eu de test à grande échelle. Il est conçu pour gérer un nombre illimité de nœuds avec des ressources minimales, en puissance de calcul et en mémoire, empruntées par chacun des ordinateurs reliés entre eux. Netsukuku peut ainsi être utilisé pour bâtir un réseau mondial distribué, anonyme et anarchique, distinct d'Internet, sans recourir à aucun serveur, fournisseur d'accès ou autorité de contrôle. Netsukuku est conçu pour construire un réseau « physique » qui ne se fonde sur aucun autre réseau existant. Il doit par conséquent y avoir des ordinateurs reliés « physiquement », Netsukuku créant ensuite les routes. C'est donc un **concurrent d'Internet** tel qu'on le connaît.

Zeronet
Il se place au milieu, il est basé sur la cryptographie réseau de BitTorrent et la cryptographie du Bitcoin.

Riffle
C'est le plus prometteur, annoncé l'année dernière, développé par le laboratoire du MIT, **il se dit 10 fois plus rapide que Tor et plus sécurisé**. Il pourrait très vraisemblablement détrôner Tor. C'est en pré-alpha et il n'y a pas eu de test à grande échelle mais la technologie et l'équipe derrière ce projet le rendent très convoité.

The farmer's Market
Adamflowers
Le premier datant de 2006, c'est un site du Clearnet. Il utilisait hushmail pour effectuer ses transactions. Il se revendiquait comme encrypté et sécurisé alors qu'en fait il travaillait en collaboration avec les forces de l'ordre. Il s'est déplacé du Clearnet à Tor **pour gagner en sécurité et anonymat**. C'était le premier véritable marché du Darknet avec un service à la clientèle tel un point de vente. Ils acceptaient l'argent liquide, Western Union, Pecunix, e-golder et paypal (ce qui est la pire idée). Ils enregistraient 3000 utilisateurs aux heures de pointes dans 55 pays, uniquement destinés à la vente de drogues.
La **riposte gouvernementale** a mis fin aux activités du site en 2012 lors de l'opération « AdamBomb » menée par la DEA avec les autorités Colombiennes, Hollandaises et Écossaises. Le chiffrement avec Tor n'était pas fautive, seulement l'utilisation de Paypal et Hushmail qui était un honeypot. Les autorités avaient ainsi accès à toutes les transactions et messages, ainsi que les déplacements d'argent. De plus, une collaboration avec les services de livraison a permis de prouver la présence de drogue dans les livraisons.

Atlantis
Apparu en 2013, c'est le **premier marché à accepter le Litecoin en plus du Bitcoin**, et n'était pas limité à la vente de drogue (clefs de produits Microsoft, tutoriels, numéros de cartes bancaires volées, etc.). Ils étaient les premiers à produire des vidéos commerciales assez comiques (https://www.youtube.com/watch?v=uD1y0kK_aH8). Ils pratiquaient également l'arnaque de confiance (les envois n'avaient pas lieu mais l'argent était retiré).

Silk Road
Premier marché moderne avec un service d'intermédiaire et une notation des vendeurs sur la qualité de leurs produits, les temps de livraison, etc. Une première version du site a été fermée par le FBI en octobre 2013, mais une nouvelle version du site est de nouveau disponible quelques semaines plus tard par une autre personne sous le même pseudo Dread Pirate Roberts, avant d'être à nouveau fermée par le FBI le 6 novembre 2014. Ils utilisaient uniquement le Bitcoin et donc l'anonymat n'était pas total.

Et alors dans le futur ?
Dark Wallet
C'est un nouveau type de portefeuille pour le Bitcoin qui implémente les paiements furtifs. Il propose également les fonctionnalités de multi-signatures, la possibilité d'utiliser un tiers de confiance et des transactions obfusquées pour l'analyse des transactions. C'est actuellement en pré-alpha.

Dash
Anciennement « Darkcoin ».

Particl
Une plateforme open source, décentralisée et anonyme et représente le futur des marchés du Darknet avec notamment la levée de 750 000 dollars. Très similaire à Etsy, mais avec une messagerie, des transactions et portefeuille cryptées dans une seule plateforme.

Zcash
Très orienté sur l'anonymat.

US Rule 41
Il s'agit d'une **loi américaine** récemment modifiée, autorisant l'état américain à s'introduire dans l'ordinateur de n'importe quelle personne, dans n'importe quel pays, dès lors que l'ordinateur est suspecté d'appartenir à un botnet. Le Royaume-Uni a récemment fait un **changement similaire**.

CONCLUSION
J'espère que cet article vous éclairera un peu sur les darknets et leurs liens avec la cybercriminalité. Le but était de partir des années 70 à aujourd'hui mais en analysant la cybercriminalité sous l'œil purement financier (on ne parlait pas de défacement de site). Finalement, on peut se rendre compte que, dans les divers scénarios de vol ou d'usurpation, les mêmes enjeux sont toujours présents : argent, drogue et sexe. A part les outils, **pas grand-chose n'a changé depuis presque 50 ans.** •

Benoit Renault
Développeur, Smile Lab

Plateformes IoT : l'open source montre la voie

Les objets connectés se multiplient et, avec eux, les firmwares, environnements systèmes et moyens de connexion. Un défi technique consiste à les intégrer là où ils n'ont pas été prévus initialement.

Comment adapter les objets connectés à chaque environnement professionnel ? La réponse passe par une plateforme d'intégration qui peut être propriétaire ou open source. Afin de suivre l'évolution rapide des standards et des protocoles de communication et pour faciliter la montée en charge dès la conception (et ainsi masquer aux métiers la complexité de l'intégration des objets connectés), la voie open source reste la mieux adaptée.
En effet, soutenue par une communauté active, la plateforme libre peut rapidement prendre en compte de nouveaux standards. Elle peut s'appuyer sur des solutions techniques éprouvées (des composants open source interchangeables) pour répondre aux problématiques de scalabilité. Enfin, depuis plusieurs années, les communautés open source démontrent leur dynamisme en termes d'innovation sur toutes les couches de logiciels, et aussi en termes de développement d'outils simplifiant le travail collaboratif des développeurs. Dans cet article, nous présentons une liste de critères permettant d'évaluer les plateformes open source et une comparaison de l'état de l'art de plateformes IoT actuellement disponibles. Ensuite, nous vous proposons de prendre en main la nouvelle plateforme CHOReVOLUTION, capable d'associer des web services et des objets connectés hétérogènes pour générer de nouveaux services sécurisés et évolutifs.

Comment sélectionner une plateforme IoT open source

Cet article composé à partir d'une étude originale et complète réalisée par Benoît Renault en août 2017, rédigée en anglais et consultable sur : https://l.ow2.org/3u.
Les critères évalués pour la réalisation de cette étude comparative incluent d'une part des paramètres génériques d'évaluation des logiciels open source (maturité du projet, support, documentation, interface utilisateur) et, d'autre part, des critères spécifiques au secteur de l'internet des objets (sécurité, connectivité, et scalabilité). Nous avons appliqué ces critères à 8 plateformes IoT considérées comme étant l'état de l'art : Sitewhere (www.sitewhere.org/), Kaa (www.kaaproject.org/), DeviceHive (https://devicehive.com/), ZettaJS (www.zettajs.org/), Parse (http://parseplatform.org/), Blynk (www.blynk.cc/), Kapua (www.eclipse.org/kapua/), NodeRed (https://nodered.org/).
L'étude a été menée dans une contrainte de temps ne permettant pas une analyse approfondie et des tests effectifs de chaque projet. Les comparatifs reposent sur l'étude des sites officiels, des documentations et codes de chaque projet. Nous invitons les lecteurs à lire le rapport original qui justifie de manière argumentée chaque score attribué.
Cette étude a été réalisée dans le cadre du projet OCCIware (http://www.occiware.org), projet de R&D financé par le Fonds National pour la Société Numérique (FSN). Les outils développés par OCCIware, fondés sur le standard OCCI, facilitent la modélisation, la conception et le déploiement cloud de toutes ressources informatiques « à la demande », IoT inclus. Ils réduisent ainsi les frontières entre les différents niveaux IaaS, PaaS et SaaS.

Maturité des projets

Solution	Première version	Dernière version	Dernier commit	Nombre de contributeurs principaux	Ratio "Open Issues"
Sitewhere	6 mars 14	19 juin 17	11 aout 17	1	0.0534653465
Kaa	30 juil. 14	28 oct. 16	31 juin 17	10	0.0173032153
DeviceHive	19 sept. 13	4 août 17	17 août 17	6	0.3434343434
ZettaJS	23 juil. 14	7 juil. 17	7 juil. 17	3	0.2565789474
Parse	28 janv. 16	3 juil. 17	18 août 17	4	0.0742424242
Blynk	9 juin 15	7 août 17	15 août 17	2	0.0295420975
Kapua	28 avr., 17	11 août 17	21 août 17	8	0.335243553
NodeRed	16 oct. 17	23 juil. 17	21 juil. 17	2	0.1063829787

(Note : Les dates présentées dans ce tableau doivent être prises en considération en tenant compte de la date de publication du rapport d'origine, en août 2017, certaines données ont pu évoluer).

Un projet peut être considéré comme mature lorsqu'il a été développé de façon la plus ininterrompue possible pendant une longue période, démontrant ainsi une stabilité et une position de pionnier. Il est par ailleurs important que le cycle de publication des versions du code soit régulier et fréquent, et que le projet affiche une date assez récente de dernier « commit » sur le code.
Le ratio « bugs ouverts » est égal au nombre de bugs ouverts divisé par le nombre total de bugs. Un ratio bas va dans le sens d'un bon niveau de réactivité et une rapidité de réponse aux contributions des utilisateurs, que ce soit pour des bugs ou des propositions d'amélioration.
Concernant le nombre de contributeurs principaux, celui-ci doit être idéalement plutôt élevé. Une situation de contributeur unique doit entraîner une certaine prudence vis à vis du projet. Il est toutefois bon d'avoir un ou quelques contributeurs principaux (idéalement plusieurs) ayant une participation conséquente dans le

répertoire du code. Comme l'indique le tableau, la plupart des projets ont démarré autour de la même période, en 2013/2014. C'est le cas également pour Parse, qui existait avant 2016 en version propriétaire ; et également pour Kapua qui existe depuis 2011 en version propriétaire et seulement depuis 2016 en open source.
Seul Blynk est arrivé un peu plus tard.
Dans l'ensemble, ces projets sont en bonne santé, comme le montrent les dates de dernière version et les dates de « commit ». A noter toutefois le cas de Kaa, qui a stoppé son cycle habituel de sortie de versions pour la préparation de la sortie de sa version 1.0 en fin d'été 2017.
Kaa, DeviceHive et Kapua sont développés par de larges équipes, alors que les autres projets sont maintenus essentiellement par des groupes de 2 à 3 personnes. De ce point de vue, la solution SiteWhere est moins performante car elle affiche un développeur unique en contributeur. Le lien de dépendance paraît évident et présente un risque significatif pour la pérennité du projet.
DeviceHive, ZettaJS et Kapua ont un ratio assez élevé de « bugs ouverts », pour différentes raisons : alors que dans le cas de Kapua, ce ratio s'explique par le fait que le projet est toujours dans sa première phase de développement en tant que projet open source, pour DeviceHive et ZettaJS cela semble davantage lié à un manque de réactivité de l'équipe de développement.
Pour résumer, les projets en meilleure position sur ce critère de la santé et de la maturité sont Kaa, Blynk, Node-Red.

Support d'entreprise

Solution	Support fourni	Adoption du projet par d'autres entreprises
Sitewhere	excellent	bon
Kaa	bon	moyen
DeviceHive	bon	bon
ZettaJS	bon	moyen
Parse	insuffisant	bon
Blynk	excellent	excellent
Kapua	bon	bon
NodeRed	bon	excellent

Le support de l'entreprise peut être considéré comme « bon » voire « excellent » s'il existe une entreprise dédiée qui supporte le logiciel et fournit un soutien aux clients utilisateurs. Si d'autres organisations utilisent ce même logiciel, cela va en faveur d'un bon niveau de fiabilité du projet. Un score « moyen » ou « insuffisant » sur ce critère « adoption » signifie que seul un petit nombre d'entreprises utilisent le logiciel.
Tous les projets bénéficient d'un support d'entreprise, à des niveaux toutefois différents, allant de l'entreprise dédiée (score « excellent ») à la grande entreprise offrant un large nombre d'autres solutions. Parse est un projet en perte de vitesse manifeste, car l'entreprise soutenant le projet n'existe plus. Ce projet est maintenant entièrement géré et dirigé par la communauté, ce qui, au vu de la chute d'activité sur le dépôt du projet, ne semble pas être un bon atout pour sa continuité.

Documentation

Solution	Documentation à jour	Adaptabilité
Sitewhere	bon	excellent
Kaa	bon	excellent
DeviceHive	bon	moyen
ZettaJS	moyen	moyen
Parse	bon	bon
Blynk	moyen	bon
Kapua	moyen	moyen
NodeRed	bon	excellent

Une documentation de qualité est un critère fondamental dans l'évaluation d'un projet open source. Ce critère a été évalué sur la base des dates de publication de la documentation, qui doivent concorder le plus possible avec les dates de publication des versions du code (documentation à jour). Une bonne documentation doit par ailleurs proposer des adaptations aux différents types d'utilisateurs (adaptabilité). Tous les projets ont une documentation relativement récente, à l'exception de ZettaJS et Kapua. Dans le cas de ZettaJS, la documentation n'a pas été mise à jour depuis longtemps et n'est pas aussi bien organisée que celle des autres projets. Quant à Kapua, l'insuffisance notée dans la documentation s'explique aisément par la récente publication en open-source du projet.

Interface utilisateur – Expérience utilisateur

Solution	Interface management de serveurs	Exemples d'applications
Sitewhere	bon	bon
Kaa	bon	excellent
DeviceHive	bon	moyen
ZettaJS	moyen	bon
Parse	bon	bon
Blynk	moyen	bon
Kapua	moyen	insuffisant
NodeRed	excellent	bon

Le logiciel doit fournir des outils appropriés à tous les niveaux. Un niveau « excellent » pour les interfaces signifie que le projet fournit des interfaces de qualité, puissantes et précises. Ce même niveau « excellent » est attribué lorsque les utilisateurs peuvent s'appuyer sur une large variété d'exemples détaillés d'applications.
En termes d'interfaces utilisateurs, tous les projets obtiennent des scores très corrects, à l'exception de Kapua, ce qui à nouveau s'explique certainement par la mise en open source récente du projet.

LES CRITÈRES SPÉCIFIQUES IoT

Sécurité

Solution	Déclarations	Audits
Sitewhere	moyen	non-existant
Kaa	excellent	non-existant
DeviceHive	insuffisant	non-existant
ZettaJS	insuffisant	non-existant
Parse	bon	non-existant
Blynk	moyen	non-existant
Kapua	moyen	non-existant
NodeRed	bon	insuffisant

La sécurité est une des caractéristiques fondamentales des systèmes IoT et doit être présente à tous les niveaux : connectivité, stockage, mises à jour, authentification, gestion des outils appropriés. L'appréciation du niveau de sécurité d'une solution est complexe et longue à évaluer, car elle nécessite un véritable audit de sécurité par des professionnels, ce qui n'est pas le cas de l'auteur du rapport. La notation a été fondée ici sur l'évaluation du niveau d'engagement de l'entreprise sur les problématiques liées à la sécurité et sur la présence d'audit de sécurité.

Le critère « Déclarations d'engagement » a été noté en fonction de l'importance et du nombre de références liées à la sécurité dans la documentation. Ce nombre permet en effet d'évaluer le niveau d'implication et de compréhension des problématiques liées à l'IoT. L'existence d'audits de sécurité aussi détaillés que possible, réalisés par des organismes professionnels indépendants, est aussi une bonne indication de l'application des bonnes pratiques liées à la sécurité.

DeviceHive et ZettaJS ne donnent que peu d'indications sur les procédures de sécurité qu'ils implémentent. SiteWhere, Blynk et Kapua donnent des détails sur l'architecture de leur sécurité. Ces détails ne sont toutefois pas à la hauteur de ceux fournis par Parse ou Note-Red. Kaa démontre de son côté un intérêt certain pour les problématiques liées à la sécurité et est définitivement la solution documentant et mettant le plus en avant son souci de la sécurité.

Aucune des solutions cependant n'a été soumise jusque-là à un audit de sécurité pratiqué par une entité indépendante.

Connectivité/Flexibilité

Solution	Plateformes Hardware	Protocoles supportés	Implémentations SDK	Modularité
Sitewhere	4	7	2	bon
Kaa	8	1	5	excellent
DeviceHive	2	3	2	moyen
ZettaJS	11	1	1	bon
Parse	3	1	7	insuffisant
Blynk	400+	2	2	insuffisant
Kapua	3	1	0	insuffisant
NodeRed	17+	5	2	excellent

Tout système IoT doit pouvoir connecter de nombreux éléments très différents, donc fournir un bon niveau de connectivité et de flexibilité. Les paramètres mesurés pour la réalisation de l'étude incluent :

- Le nombre de plateformes hardware compatibles
- Le nombre de protocoles supportés, tels que MQTT, AMQP, OpenWire, XMPP, HTTP REST, etc.
- Le nombre d'implémentations du SDK (facilite l'intégration de la solution avec le langage préféré du développeur)
- La modularité (capacité de changer certains composants)

Nous recommandons, pour l'aspect connectivité, de lire l'étude détaillée du projet pour mieux comprendre les scores attribués (à télécharger sur https://l.ow2.org/3u). En effet, la connectivité d'une solution est un critère difficile à évaluer. Chaque solution fournit son propre moyen de communication avec les logiciels ou appareils externes, avec leur propre vocabulaire et paradigmes.

En termes de connectivité, les solutions les plus performantes sont Blynk pour sa compatibilité avec les plateformes hardware, SiteWhere pour son support d'un grand nombre de protocoles de communication, Kaa et Parse pour le nombre d'implémentations du SDK proposé et Node-Red et Kaa pour leur excellente modularité. Dans l'ensemble, Kaa semble être la meilleure option en termes de connectivité et de modularité.

Scalabilité

Solution	Scalabilités des projets tiers	Scalabilité de la plateforme
Sitewhere	bon	insuffisant
Kaa	excellent	excellent
DeviceHive	excellent	moyen
ZettaJS	insuffisant	insuffisant
Parse	moyen	insuffisant
Blynk	moyen	insuffisant
Kapua	moyen	insuffisant
NodeRed	bon	bon

Les systèmes IoT sont amenés à croître très rapidement. Il est donc nécessaire que l'application permette un passage à l'échelle rapide et simple, offrant la possibilité à un grand nombre d'outils dispersés dans le monde entier de se connecter au système. On distingue deux types d'évolutivité :

- La scalabilité des composants : les solutions IoT open-source résultent souvent d'une combinaison de plusieurs sous-projets open source. Les scores attribués sur ce critère dépendent donc de la capacité des sous-projets concernés à être eux-mêmes évolutifs.
- La scalabilité de la plateforme : les solutions IoT sont intégrées dans des systèmes plus larges, des environnements d'exécution. Ce score est fixé en fonction de la capacité à proposer des solutions distribuées au travers de différentes machines tout en maintenant une communication entre ces solutions.

Bien que la plupart des solutions présentées ici se présentent comme scalables, seulement certaines d'entre elles justifient, d'une façon claire, comment elles y parviennent. Alors que Node-Red et DeviceHive semblent apporter une réflexion sur la question, seule Kaa présente une réelle approche vers la scalabilité, avec la distribution de données intégrées, la communication inter-instances, et l'équilibrage de charges.

En conclusion, cette analyse montre qu'il existe un certain nombre de plateformes IoT open-source très correctes. Certaines d'entre elles sont déjà utilisées en production, en grand partie par les entreprises qui les développent. Chacune de ces solutions présente une approche différente et des fonctionnalités spécifiques, qui ne sont pas toutes décrites ici. Les solutions fondées sur NodeJS sont très intéressantes, en particulier Node-RED qui amène un niveau d'abstraction graphique particulièrement remarquable. Kapua semble être une solution très prometteuse pour le futur, même si le projet souffre pour l'instant d'un manque apparent de maturité. DeviceHive et SiteWhere apparaissent également comme des solutions très satisfaisantes, bien que le manque de documentation sur les précautions liées à la sécurité pour DeviceHive et le manque de scalabilité pour SiteWhere les rendent moins attrayantes pour le moment.

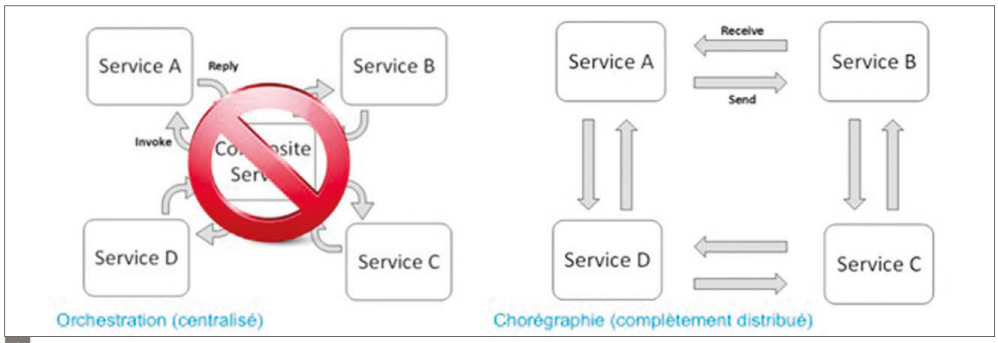
Pour lire le rapport détaillé (en anglais) : https://l.ow2.org/3u

Après avoir apporté un panorama de certaines solutions IoT existantes, mettons maintenant l'accent sur une plateforme plus globale, et issue de trois années de R&D, la plateforme CHOReVOLUTION. •

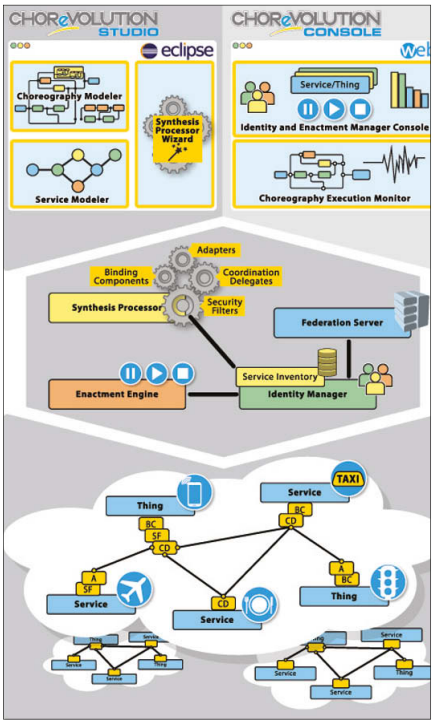
Sébastien Keller
R&D Manager, Thales

CHOReVOLUTION,

UNE PLATEFORME OPEN SOURCE POUR DÉVELOPPER RAPIDEMENT DE NOUVEAUX SERVICES INTÉGRANT DES OBJETS CONNECTÉS.



1



Comment, aujourd'hui, intégrer les services existants, les objets connectés pour en faire de nouvelles applications ? Ces dernières années, cette approche a été portée par les intégrateurs qui, au sein d'une organisation, combinaient les services existants pour offrir de nouvelles applications au moyen de workflows élaborés. Au niveau architecture, des solutions complètes et performantes ont été proposées en commençant par les offres EAI (Enterprise Application Integration), suivies par les solutions ESB (Enterprise Service Bus). Toutes ces approches avaient en commun la mise en place d'un nouveau composant serveur qui gérait toutes les interactions. Certes, certains fournisseurs ont proposé une architecture distribuée pour ce composant serveur, néanmoins son rôle central reste un frein pour des intégrations à grande échelle qui ne veulent pas fixer une limite au nombre de services/ objets connectés intégrés.

Comment aller plus loin ? Comment éviter de passer par un composant centralisé qui limiterait la scalabilité de la nouvelle application ? Comment rendre dynamique ces intégrations entre services/ objets connectés ? Comment gérer la sécurité de l'ensemble ? Comment intégrer des services ou des objets connectés gérés par des fournisseurs différents ayant des règles de sécurité différentes ? La plateforme open source CHOReVOLUTION, grâce à ses composants, fournit une alternative intéressante s'appuyant sur le concept de chorégraphie de services, là où les solutions habituelles utilisent principalement le concept d'orchestration.

La plateforme CHOReVOLUTION adresse l'ensemble de la chaîne de développement, incluant la conception, le déploiement de services distribués et la supervision d'applications IoT. Elle est issue d'un projet R&D Européen Horizon 2020 qui regroupe huit partenaires, coordonnés par Thales. CHOReVOLUTION offre aux développeurs un cadre de travail complet, partant des processus métiers jusqu'au déploiement de services dans le cloud. Elle se positionne donc sur le segment de marché des solutions de génération d'applications IoT (ou IoT Applications Enablement Platforms).

Le principe de chorégraphies de services par opposition à l'orchestration 1

Habituellement, l'intégration de services en vue de créer une nouvelle application ou un nouveau workflow s'appuie sur le principe d'orchestration. Dans ce cas, un composant dédié gère de manière centralisée, toutes les interactions entre les services et/ou les objets connectés à la manière d'un chef d'orchestre. Une chorégraphie va s'appuyer davantage sur la définition d'un ensemble d'étapes qui seront toujours exécutées de la même manière, indépendamment de l'orchestrateur. L'intégration de ces chorégraphies va donc donner lieu à un ensemble d'exécutions séparées (dans le temps et dans l'espace) où seuls des points de synchronisation permettront à l'ensemble d'atteindre un objectif global.

Le principe de chorégraphie présente deux principaux avantages qui sont, premièrement, d'aller vers des architectures complètement distribuées et, deuxièmement, de limiter les accès aux services aux seuls acteurs impliqués dans les chorégraphies. Dans le cas d'une orchestration, tous les services doivent autoriser l'orchestrateur à les appeler.

Une plateforme globale, couvrant l'ensemble du cycle de développement

Une autre particularité de la plateforme CHOReVOLUTION est de fournir une palette d'outils associés à un processus couvrant l'ensemble du cycle de développement d'applications (appelé SDLC, « Software Development Life Cycle »). Ce cycle commence par l'enregistrement des services et objets connectés existants et se poursuit par la création de chorégraphies correspondant aux différentes interactions entre ces services et objets. Ces chorégraphies sont modélisées au moyen du standard BPMN2. A l'issue des modélisations, le concepteur d'application peut décider de développer de

nouveaux services et/ou de déployer de nouveaux objets connectés afin de pouvoir réaliser l'ensemble des chorégraphies. Ensuite il va relier ces modélisations aux services et objets connectés réels. A ce stade, il a terminé la conception de la nouvelle application. Concernant la partie développement, la plateforme CHOReVOLUTION génère ensuite l'ensemble des composants exécutables. Après cette phase de développement, le déploiement des composants est organisé ainsi que la configuration associée. Les deux parties suivantes présentent la plateforme elle-même ainsi qu'un exemple de création d'application en se focalisant sur la partie conception.

Les trois parties complémentaires de la plateforme

La plateforme est constituée de trois parties distinctes. Ce sont les composants frontaux (frontend), manipulés par les utilisateurs eux-mêmes, les composants serveurs (au niveau backend), utilisés par les composants frontaux et aussi lors de l'exécution des chorégraphies, et, enfin, la plateforme d'exécution sur laquelle tous les composants sont déployés et connectés avec les services/objets réels.

Les composants frontend :

Les deux composants «frontend» sont, d'une part, l'atelier permettant de concevoir, développer et générer les chorégraphies exécutables ; cet atelier est fondé sur l'IDE Eclipse. Et, d'autre part, des consoles permettent de surveiller l'exécution des chorégraphies et la gestion des différents services/objets connectés. Ces consoles sont fournies au travers d'applications Web.

Les composants serveurs :

La plupart de ces composants sont utilisés pendant les phases de conception et également pendant les phases d'exécution. Le premier de ces composants est le «Synthesis Processor» qui assure la génération des chorégraphies exécutables qui sont produites à l'issue de la conception ou pendant l'exécution lors d'un redéploiement partiel. Les deux composants suivants sont le «Federation Server» qui gère la sécurité et l'«Identity Manager» qui contribue également à la sécurité et à l'enregistrement des services et objets connectés réels. Ces deux composants sont fondés sur le projet Apache Syncope. Le dernier composant, l'«Enactment Engine», configure et organise le déploiement des chorégraphies exécutables.

La plateforme d'exécution

La plateforme d'exécution est une plateforme Cloud OpenStack qui est utilisée pour déployer les chorégraphies exécutables. Elle est connectée avec l'extérieur pour s'interfacer avec les services et les objets connectés réels. Suivant les contextes d'exécution, cette plateforme peut être une plateforme déjà existante. Dans ce cas, les outils de monitoring ne peuvent être les mêmes que si la plateforme existante est compatible avec la plateforme CHOReVOLUTION initiale.

CHOReVOLUTION en action : création d'une application e-commerce

L'exemple que nous allons prendre appartient au domaine e-Commerce, il concerne l'achat de différents produits de différentes natures. Dans notre exemple, le client est connecté à Internet au moyen d'un ordinateur ou d'une application mobile. Il peut sélectionner une liste de produits et, dès que le paiement a été réalisé, la commande peut être exécutée et la facture envoyée au client. Suivant la date de livraison annoncée, les produits sont conditionnés et envoyés au client qui reçoit alors les informations d'expédition. 2

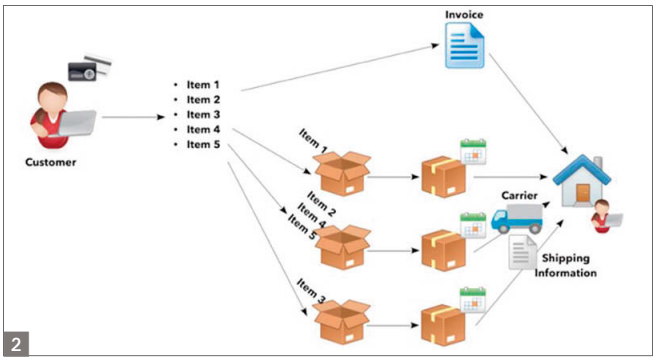
Pour réaliser rapidement cette application avec la plateforme CHOReVOLUTION, les principales étapes à suivre sont :

1. Les déclarations de services/objets à utiliser,
2. La conception de chorégraphies combinant les services et les objets,
3. La génération de chorégraphies exécutables,
4. Le déploiement des chorégraphies,
5. L'exécution et la surveillance des chorégraphies.

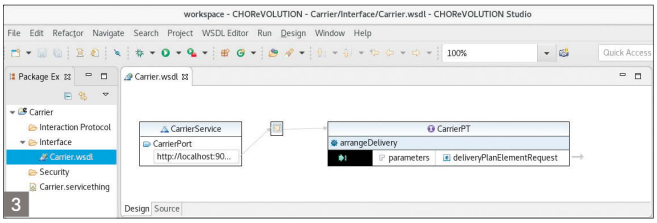
Nous allons dans cet article, décrire seulement les deux premières étapes. L'ensemble de ces étapes sont décrites dans le tutoriel en ligne (https://l.ow2.org/cvms).

Comment démarrer avec la plateforme CHOReVOLUTION ?

Le meilleur moyen de démarrer est de suivre le programme de la campagne de betatest de CHOReVOLUTION accessible sur le site web du projet (https://l.ow2.org/cpvm) Au travers de ce programme, vous serez guidé pour télécharger VirtualBox, télécharger la machine virtuelle (VM) contenant les différentes parties de la plateforme CHOReVOLUTION et suivre le « Getting started » pour construire une nouvelle application.



2



3

N'hésitez pas à suivre cette campagne de betatest, elle peut vous permettre de gagner un drone !

Déclarations des services/objets à utiliser :

Le rôle de cette étape est de définir les services que nous voulons voir interagir. Dans cette partie, nous allons montrer comment importer un service (le service «Carrier»). Pour cela, démarrer le studio CHOReVOLUTION qui est fondé sur Eclipse et effectuer les actions suivantes (actions déjà réalisées dans la VM) :

- Choisir File -> New -> CHOReVOLUTION Service/Thing Project.
- Saisir un nom de projet, par exemple "Carrier" et laisser les valeurs par défaut, cliquer sur Next ;
- Sélectionner Web Service Description Language et cliquer sur Finish.

Le projet « Transporteur » est créé et ouvert dans l'explorateur de packages.

- Importer la description du service (fichier Carrier.wsdl), pour cela glisser-déposer le fichier (présent dans l'archive téléchargée) directement dans le répertoire des interfaces. On obtient la description suivante : 3
- Avec la souris, cliquer sur du bouton droit sur le fichier «Carrier.servicething» et choisir CHOReVOLUTION Tools -> Upload Service into Service Inventory. L'Assistant «Service Inventory» est affiché : 4
- Spécifier les différentes valeurs comme suit (actions déjà réalisées dans la VM) :
  - Service endpoint, avec http://localhost/carrier/carrier ;

- Service Role name, avec « Carrier_role ».

Et cliquer sur Create puis sur OK. Vous obtenez l'écran suivant afin d'enregistrer le service : 5

- Sélectionner un rôle spécifique pour ce service dans la liste Available roles, «Carrier_role» et cliquer sur Add.
- Cliquer sur OK dans la boîte de dialogue s'affichant.

Le service «Carrier» est maintenant présent dans l'annuaire de services que nous voulons utiliser pour créer notre application. En suivant la même démarche, il faudrait importer les descriptions des autres services en créant les différents CHOReVOLUTION Service/Thing Project pour les services Invoicer, Scheduler, Payment System afin de les référencer dans notre annuaire de services (actions déjà réalisées dans la VM).

Conception des chorégraphies combinant les services et les objets

Le rôle de cette étape est de créer les chorégraphies. La première étape consiste à créer un projet spécifique et ensuite, dans une seconde étape, à construire la chorégraphie. Commençons par démarrer le studio CHOReVOLUTION et effectuer les actions suivantes :

- Choisir File -> New -> CHOReVOLUTION Synthesis Project,
- Saisir un nom de projet (par exemple " GettingStartedCaseStudy" et laisser les valeurs par défaut, cliquer sur Next et cliquer sur Finish.(*)

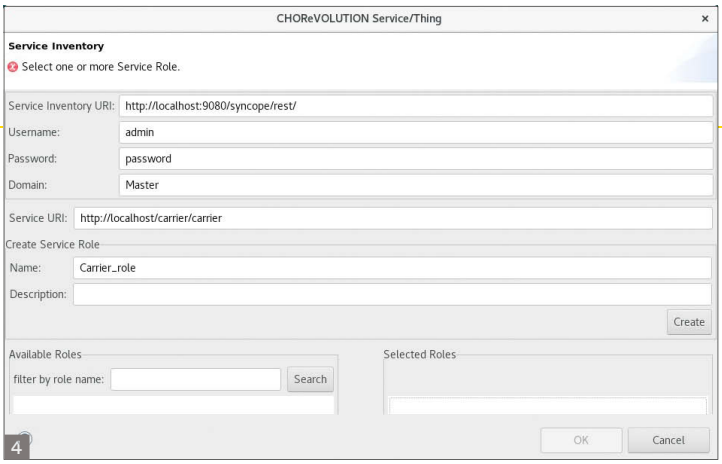
Vous êtes maintenant prêts à créer une chorégraphie pour notre cas d'utilisation. Pour cela un composant Eclipse permettant de les modéliser avec le langage BPMN2 est fourni. Pour obtenir plus de précisions sur l'utilisation de ce composant, n'hésitez pas à consulter la documentation en ligne (téléchargeable à https://l.ow2.org/2c).

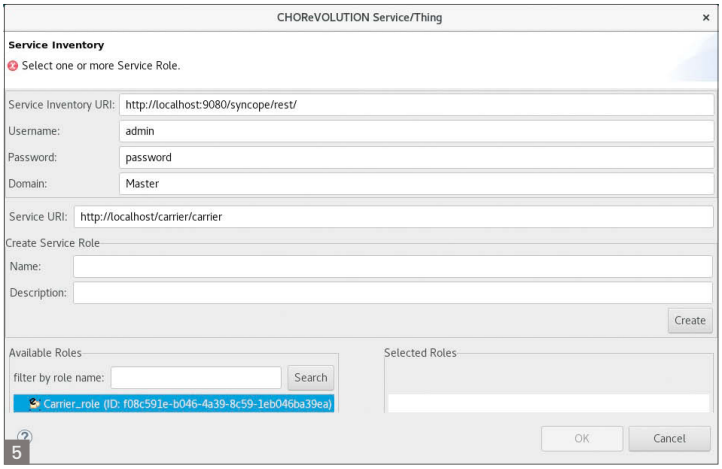
- Sélectionner le répertoire «Choreography Diagrams» et avec le bouton droit de la souris, choisir New -> Other -> CHOReVOLUTION -> BPMN 2.0 Choreography Diagram. Cliquez ensuite sur Next.
- Saisir un nom de fichier, « CHOReVOLUTION-GettingStarted » dans le champ File name et cliquer sur Finish.

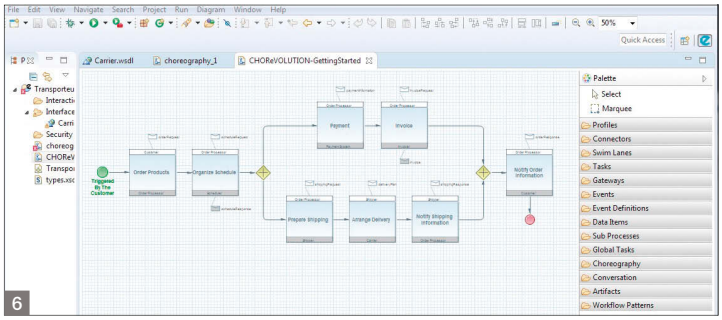
Vous obtenez un diagramme vide qui est à compléter en utilisant la palette d'outils qui se trouve sur la gauche. Dans la VM téléchargée, vous pouvez accéder au résultat en double cliquant sur le fichier *getting-started.bpmn2*, présent dans le répertoire *Choreography Diagrams*. 6

(*) Note

Noter que pour obtenir une génération correcte des futures chorégraphies exécutables, vous devez utiliser le JDK Java et non le JRE.







Les étapes suivantes :

Une fois le modèle BPMN2 de la chorégraphie dessiné, les différents composants de la plateforme permettent de générer des chorégraphies exécutables et de les déployer dans une infrastructure cloud. La génération des chorégraphies exécutables s'effectue au travers des trois dernières étapes. Ces étapes sont décrites dans le « Getting started » dans la partie «Stage Three: Start the Synthesis Process » et peuvent être effectuées avec la VM téléchargée. Le déploiement est réalisé au moyen d'un fichier décrivant l'ensemble des éléments à déployer ainsi que les contraintes et les configurations associées.

En conclusion, la plateforme CHOReVOLUTION fournit un bon exemple de ce qu'offre une plateforme IoT open source. Elle se distingue en termes d'automatisation et de qualité de développement, grâce à une approche originale centrée sur la chorégraphie de services. Son objectif reste de contribuer à l'innovation, au travers de nouvelles applications évolutives et sécurisées. En résumé, cette plateforme s'inscrit complètement dans l'approche DevOps et dans la transformation numérique des entreprises. La plateforme est partagée en open source depuis le début de sa conception. Elle bénéficie de l'accompagnement d'OW2 qui contribue à offrir de bonnes pratiques de développement, d'organisation et de gouvernance de composants open source, pour en faire un produit mature et utilisable par des acteurs de l'industrie. La plateforme fait maintenant l'objet d'un programme de beta tests, de manière à recueillir des avis et propositions d'amélioration de ses composants. Un atelier, dédié à l'utilisation du studio CHOReVOLUTION, a été organisé à l'occasion du salon Paris Open Source Summit, le 6 décembre 2017 après-midi. •

• Florent Santin
Mode bidouille
fsantin@infinitesquare.com

Atelier bricolage : créer sa borne d'arcade !

Dans un style complètement différent des articles de programmation, je vous propose de partager mon retour d'expérience sur la conception d'une borne d'arcade maison.

Etant né dans les années 80, j'ai passé une partie de mon enfance à dépenser mon argent de poche dans des bornes d'arcades et dans des flippers. Devenu grand garçon, cela fait quelques années que l'envie soudaine de posséder ma propre borne me trotte dans la tête. Pourquoi ? Pour plein de raisons : retrouver les sensations de mon enfance, pouvoir rejouer avec mes potes à des jeux que je n'ai jamais pu finir, en découvrir d'autres et surtout, partager ces sensations avec mes enfants (et au passage récupérer mon investissement en les rackettant avec un monnayeur, nous y reviendrons).

Ok, avoir une borne, c'est chouette, mais comment s'en procurer une ?

Il y'a trois façons de faire :

- Racheter une vieille borne d'occasion et la retaper complètement : c'est la meilleure approche pour les puristes qui souhaitent retrouver les bonnes sensations. Seul souci, il faut quand même avoir beaucoup de temps et quelques compétences en électronique que je n'ai pas.
- Acheter une borne d'arcade neuve : des entreprises créées par des sociétés vendent aujourd'hui des bornes d'arcade de style retro, en partant de composants neufs et industriels, avec tout le service de personnalisation et de support. A titre d'exemple, la société Neo Legend (www.neo-legend.com), basée à Paris, propose des bornes de qualité. J'ai longuement hésité, mais il me manquait dans ce scénario le côté « c'est moi qui l'ai fait ».
- Et la troisième manière, celle que j'ai finalement choisi : partir de rien, et tout construire !

Par où commencer ?

La première chose à faire, c'est se documenter. J'ai de mon côté passé plusieurs heures en recherche de site en site, pour finalement trouver les pointeurs intéressants. Pour ceux qui veulent se lancer, sur la construction, je ne peux que conseiller le blog « mister bidouilles » dont je me suis très fortement inspiré (http://blog.mister-bidouilles.fr/construction-d-une-borne-d-arcade-etape-1/) pour démarrer.

Combien de temps cela va prendre ?

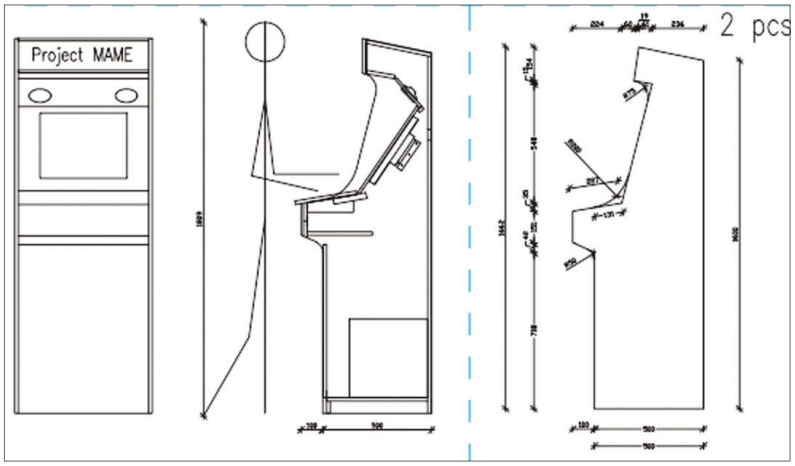
De mon côté, il s'est écoulé 2 ans entre le début et la fin. Bon, comme beaucoup de développeurs, j'ai plein de motivation mais une fâcheuse tendance à commencer quelque chose rapidement (découper le bois – 1 journée de travail) puis à mettre tout en suspens avant de retrouver la motivation et terminer. De manière pragmatique, il m'aura fallu environ 3 semaines intensive de travail en soirée pour finir, dont une bonne partie en binôme avec ma femme (support psychologique + assemblage nécessitant d'être deux + peinture).

La construction

L'avantage en construisant tout sur mesure est que vous pouvez adapter la taille de votre borne à l'emplacement dans votre intérieur. Les bornes étaient très volumineuses dans le temps notamment à cause de la grande taille des écrans cathodiques mis à l'intérieur, accompagnés d'une électronique complexe pour le système de jeux. En 2018, un écran plat et un Raspberry permettent de réduire au strict minimum le volume nécessaire pour notre borne. Le seul choix du modèle est donc purement esthétique.

De ce fait, vous allez surtout trouver deux types de plans :

- Les plans pour borne d'arcade tradition-



nelle, pour une borne d'environ 1,80m de hauteur, et un carré de 0,60m de largeur / profondeur.

- Les plans pour borne d'arcade réduite, dite « bartop », pour une hauteur de 0,60m uniquement. Il s'agit de bornes qui ne contiendront que le panel de jeu et l'écran, sans tout l'encombrement du support, et à poser donc directement sur un meuble ou une table.

Quel que soit le type de borne, vous pouvez trouver des plans et des pas à pas de montage détaillés sur le site http://koenigs.dk/mame/ (en Anglais).

De mon côté, même si encombrant et partant du principe que la borne sera un élément décoratif de mon intérieur, j'ai opté pour un grand format, afin de pouvoir jouer debout et être le plus proche possible des bornes des années 80.

En termes de bois utilisé, deux possibilités également :

- Effectuer de la récupération en découpant des vieux meubles. Peu couteux, mais très chronophage, notamment sur les phases de peinture car il va falloir poncer les vieilles planches.
- Utiliser du bois neuf. Le bois idéal à travailler est du MDF (Medium Density Fiberboard). Il est vendu dans toutes les grandes enseignes de découpage.

J'ai, de mon côté, opté pour l'achat de bois neuf. Les panneaux les plus importants sur la borne sont les deux panneaux latéraux, le socle, et le haut. Ce sont eux qui vont constituer l'armature globale et tenir l'ensemble du poids, d'où l'utilisation de MDF assez épais (18mm). Les autres panneaux, avant (monnayeur, support Joypad, support haut-parleur) et arrière ne servent pas à renforcer la structure et sont purement décoratifs, du bois d'épaisseur plus faible (10mm dans mon cas) sera plus économiquement intéressant.

Ayant mes plans et pour gagner du temps, j'ai opté pour une découpe en magasin sur l'ensemble de mes planches, à l'exception des panneaux latéraux. En effet, les machines industrielles de magasin de bricolage sont adaptées pour les coupes droites, mais ne peuvent pas faire des découpes arrondies telles que nécessaires pour les panneaux latéraux, le support du Joypad et le support haut-parleur. Pour cette découpe, pas le choix : mètre, équerre, compas sont requis pour dessiner les trais de découpe avant de passer à la scie sauteuse. La difficulté est que les deux panneaux doivent être strictement identique. L'idéal est d'en découper un et de l'utiliser comme modèle pour le deuxième.

Une fois les planches découpées, peut commencer le montage. Pour réussir une belle borne, il faut réussir à faire en sorte de n'avoir aucune vis apparente sur les faces latérales de la borne. Le seul moyen de réussir ceci est d'utiliser des tasseaux de bois et de les visser de l'intérieur en utilisant de la visserie à la bonne longueur pour ne pas traverser. Une bonne astuce consiste à utiliser de la colle à bois avant de visser pour solidifier et bien positionner les éléments. Les tasseaux vont uniquement être présents sur les panneaux latéraux, qui sont épais pour les soutenir.

Une fois l'ensemble des tasseaux posés, l'assemblage peut démarrer. Les planches fines, de face et d'arrière, vont venir être collées et vissées dans les tasseaux et solidifier l'ensemble.

Etant un meuble volumineux et creux, il ne faut pas hésiter à emménager le bas de la borne pour servir de rangement, car il ne sera pas utilisé par le hardware. De mon côté, j'ai fait en sorte de pouvoir ouvrir l'arrière de la borne avant d'accéder à l'intérieur et y ranger du matériel.

En termes de coût, pour une borne grand format, il faut prévoir un peu moins d'une centaine d'euros pour l'ensemble du bois, panneaux, coût de la prédécoupe et tasseaux.

Une fois l'assemblage final terminé, la dernière action manuelle sur le bois est la préparation de la rainure sur les panneaux latéraux pour positionner les T-Moldings. Le T-Molding est une bande caoutchoutée de 20mm d'épaisseur qui va venir protéger les bords des panneaux latéraux. En plus de protéger les angles, cela apporte une finition d'une couleur différente et permet de donner un aspect arrondi.

C'est par contre très difficile à poser, car cela nécessite une rainure de quelques millimètres pour être enfoncé. Un bon bricoleur sera équipé d'une défonceuse pour faire cette rainure, qui doit être pile au milieu du bois (à 8mm de chaque bord). Dans mon cas, ne voulant pas investir dans cet outil, j'ai dû me contenter d'un cutter électrique : difficile d'être précis et bien centré, et très long à faire (4h). Une fois la rainure faite, il suffit d'enfoncer la bande

de T-Molding sur toute la longueur avec un maillet, et le tour est joué. Pour ma borne, il m'aura fallu 5 mètres de T-Molding chromé de 19mm (3/4), achetés 21 euros.
Une fois la borne assemblée, reste à faire sa décoration. Là aussi, plusieurs possibilités s'offrent à vous :

- Faire une création dans Photoshop, et faire imprimer des stickers exactement aux dimensions de la borde d'arcade, de type PVC adhésif (prévoir un budget d'environ 80 euros).
- Se sentir l'âme d'un artiste et décorer soit même avec de la peinture à bois

De mon côté, souhaitant une borne au design relativement sobre et s'intégrant parfaitement dans mon intérieur, j'ai opté pour l'option sous couche et peinture à bois. Une vingtaine d'euros de pot de peinture, quelques pinceaux et plusieurs couches plus tard, le tout était peint en blanc, avec quelques rappels noirs.

L'écran

C'était ma zone de stress, trouver un écran à mettre dans la borne.
Pour des raisons de place, j'ai opté pour un écran LCD plutôt qu'un écran cathodique. C'est raté pour l'effet pixel et balayage de l'écran, mais au moins, cela tiendra plus longtemps.
En termes de format, un écran de 19" est idéal, c'est un bon confort pour jouer à deux, et cela reste harmonieux
Le seul problème, c'est que depuis quelques années, l'intégralité des écrans de PC de ou de TV neufs sont au format 16/9eme, qui n'est pas du tout adapté au monde de l'arcade.
Pour trouver un écran plat, de 19', au format 4/3, il n'y a qu'une seule solution, chercher dans le monde de l'occasion. De mon côté, après avoir demandé à tous mes collègues « vous devez bien avoir ça chez vous » sans succès, je suis tombé sur une annonce sur Le Bon Coin – coût de l'achat : 30 euros. Il faut également prévoir un adaptateur VGA (ou DVI) vers HDMI pour brancher ensuite l'écran.
Reste ensuite à fixer et à centrer l'écran. Là aussi, des tasseaux des bois et un peu de bricolage sont requis pour créer un cadre suffisamment robuste pour supporter le poids de l'écran.
Une fois en place, il reste à habiller le tour de l'écran, afin de masquer son cadre en plastique et ses boutons pour ne laisser ressortir que l'image. De mon côté, j'ai fait découper une plaque de plexiglass transparent pour recouvrir l'ensemble de l'écran. Obligé de retourner dans un magasin de bricolage pour le faire découper à la bonne taille car c'est assez compliqué à travailler. Prévoir tout de même une vingtaine d'euro pour la plaque découpée. Pour masquer les côtés, nous avons protégé le centre dédié à l'écran avec du scotch, puis peint les contours de la plaque avec une bombe de peinture noire. Le résultat donne, lorsque la borne est éteinte, une plaque uniforme noire, et lorsque la borne est allumée, l'image apparait au milieu de cette plaque. En bonus, j'ai également ajouté 2 douilles et ampoules de 220V au-dessus de l'écran, cachées derrière une plaque de plexi opaque afin d'illuminer le haut de la borne quand celle-ci est en marche.

Le son

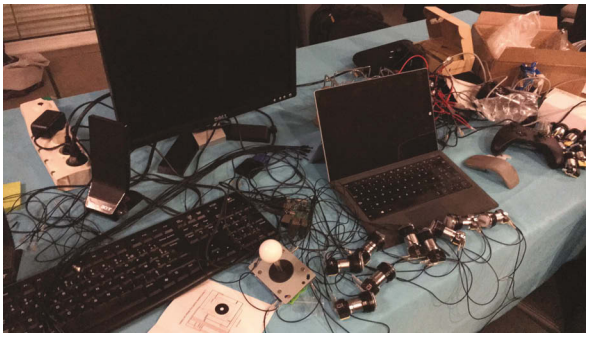
Après l'image, le son. Il est essentiel de pouvoir écouter les musiques et sons des jeux, bien qu'une restitution en full HD ne soit pas nécessaire pour jouer des sons 8 bits.
Là aussi, il y a deux possibilités :

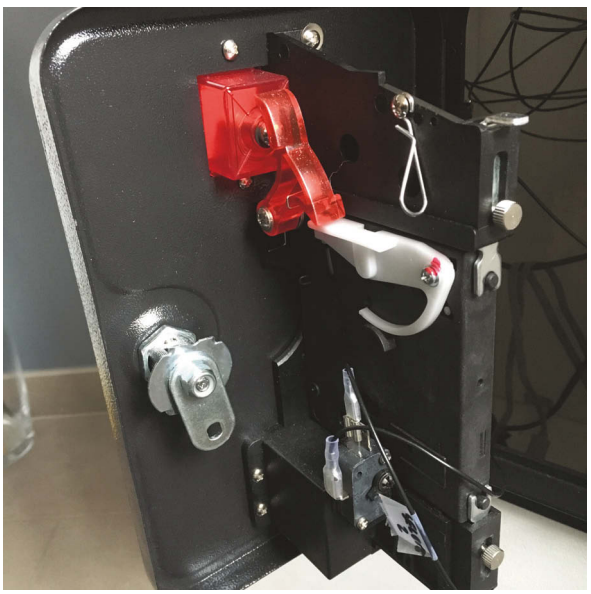
- Recycler une vieille paire d'enceinte PC, avec une prise Jack 3.5mm, et fixer le tout derrière l'écran. Cela permet d'aller vite, mais ce n'est pas pratique pour contrôler le volume, et visuellement, les enceinte ne seront pas visibles ou mal intégrées.
- Acheter un mini amplificateur audio, alimenté en 12V, un peu de câble audio, deux hauts parleurs 10w et des grilles décoratives pour mettre dessus.

Je suis de mon côté parti sur cette 2ème option, et ait percé deux trous de 80mm au-dessus de mon écran pour venir y encastrer les 2 hauts-parleurs. Les deux haut-parleurs sont ensuite reliés à l'ampli que j'ai caché sous le panneau de contrôle, avec juste un petit trou percé pour pouvoir régler discrètement le volume. Le son est envoyé depuis mon système de jeu vers l'amplificateur avec une traditionnelle sortie audio de type Jack.

Le système de jeu

Fini le bricolage, vient ensuite l'électronique. La première étape est donc de choisir le device qui servira à gérer le système de jeu.

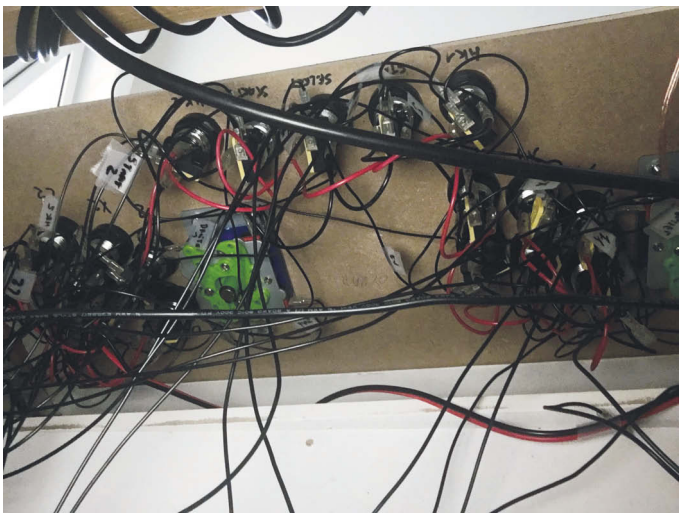


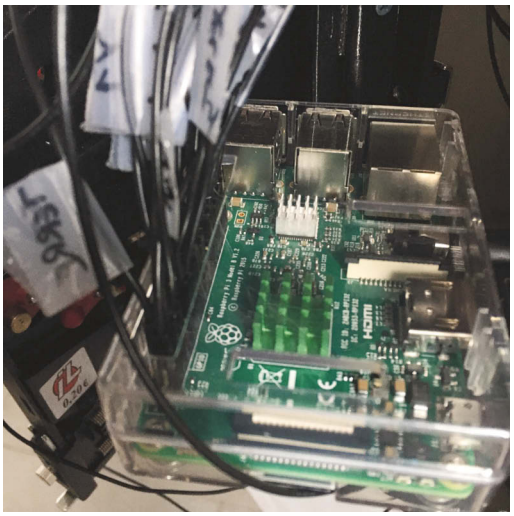


Dans mes recherches, j'ai retenu deux alternatives intéressantes quasiment prêtes à l'emploi :

- HeavyBox : il s'agit d'une configuration multi-émulateurs qui peut être installée sur n'importe quel PC et qui utilise le Frontend (l'interface de lancement des jeux) HyperSpin. La configuration s'effectue en quelques heures en suivant les bons tutoriaux et le rendu d'HyperSpin est vraiment très qualitatif.
- Recalbox (https://www.recalbox.com/) : il s'agit d'un système d'exploitation pour Raspberry PI dédié à l'émulation s'appuyant sur un OS prêt à l'emploi, RecalboxOS, lui-même basé sur du Linux.

Power 3V3	1	2	Power 5V
HotKey J1	3	4	Power 5V
HotKey J2	5	6	GND
Haut J1	7	8	Ⓛ / Btn3 J1
GND	9	10	Ⓧ / Btn2 J1
Bas J1	11	12	Ⓨ / Btn1 J1
Gauche J1	13	14	GND
Droite J1	15	16	Ⓡ / Btn6 J1
Power 3V3	17	18	Ⓐ / Btn5 J1
Start J1	19	20	GND
Select J1	21	22	Ⓑ / Btn4 J1
Haut J2	23	24	Ⓛ / Btn3 J2
GND	25	26	Ⓧ / Btn2 J2
Non utilisé	27	28	Non utilisé
Bas J2	29	30	GND
Gauche J2	31	32	Ⓨ / Btn1 J2
Droite J2	33	34	GND
Start J2	35	36	Ⓡ / Btn6 J2
Select J2	37	38	Ⓐ / Btn5 J2
GND	39	40	Ⓑ / Btn4 J2





Il y a deux ans, au début du projet, j'ai d'abord creusé en profondeur le projet HeavyBox. Son avantage est qu'il m'a permis de faire de l'émulation directement sur mon PC. Lorsque j'ai avancé dans mon projet, je me suis retrouvé face à la seule contrainte : installer un PC dans la borne. Un PC, même en récupération, est un petit investissement, cela prend de la place, cela chauffe et cela met un peu de temps à démarrer (pas pratique pour une partie rapide). Par contre, son avantage, c'est qu'il peut être très puissant notamment pour émuler des jeux en 3D.

Dans la version finale de ma borne, j'ai finalement choisi Recalbox. Avec ce système, j'ai fait une croix sur les jeux en 3D, mais j'ai gagné un vrai confort : le système tourne sur un Raspberry PI, s'installe en copiant la distribution sur une clé USB, puis, une fois démarré, est accessible en partage réseau. L'avantage du projet Recalbox est également qu'il est en constante évolution, et que la distribution est capable de se mettre à jour toute seule en se connectant au réseau Wifi. Enfin, à l'opposé du PC, le Raspberry PI ne prend pas de place, ne chauffe pas, et coûte relativement peu cher : une cinquantaine d'euros pour un Raspberry PI 3 en ajoutant l'alimentation, une carte SD de 16 Go (largement suffisant) et un boîtier de protection.

Les contrôles

Un autre avantage et non des moindres sur Recalbox / Raspberry est qu'il existe des kits prêts à l'emploi de Joysticks et de boutons, avec plan de câblage pour se connecter directement sur les GPIO du Raspberry. Le câblage est donc à la portée de n'importe qui pour venir connecter les 17 boutons et les 2 Joysticks. En termes d'achat, j'ai comme l'ensemble de mon matériel opté pour le site SmallCab, qui propose des kits allant de 65 à 84 euros. J'ai opté pour le kit qui avait la particularité de proposer des boutons chromés lumineux.

Les 17 boutons ont les rôles suivants :

- 6 boutons pour joueur 1 + 1 bouton Hotkey utilisé sur certains jeux ;
- 6 boutons pour joueur 2 + 1 Hotkey ;
- 1 bouton « Insert Coin » pour ajouter du crédit ;
- 1 bouton « joueur 1 » et 1 bouton « joueur 2 ».

Pour l'installation des boutons et des Joysticks, le plus compliqué a été de percer les différents trous dans une planche de bois, mais une fois cette opération effectuée, le câblage a été fait avec minutie en 1 heure environ. En termes de câbles à relier, cela représente :

- 4 câbles par Joystick (haut, gauche, bas, droite) x2 ;
- 17 câbles pour les boutons ;
- 2 câbles reliant l'ensemble des boutons entre eux et le Raspberry pour fermer le circuit ;
- 1 grand câble pour alimenter en 12v les LED des boutons lumineux.

Pour ajouter une petite cerise sur le gâteau, et vraiment disposer d'une borne d'arcade maison proche d'une vraie, j'ai également décidé d'acheter un monnayeur. L'investissement est assez élevé par rapport au reste, car la porte de monnayeur, en métal, coûte 55 euros auxquels il faut ajouter 15 euros pour le système mécanique qui reconnaît les vraies pièces de monnaie (20 centimes dans mon cas). Au niveau du câblage, le monnayeur est relié comme n'importe quel bouton au Raspberry et vient se substituer au bouton « Insert Coins ».

Enfin, pour pouvoir également jouer avec tout le confort de mon enfance, j'ai décidé de connecter au Raspberry 2 manettes SFC30 sans fil Bluetooth de la marque 8BitDot, achetées 30 euros pièce sur Amazon. Plusieurs avantages : elles ressemblent à des vraies manettes de Super Nintendo, elles sont sans fils, et surtout, elles sont nativement compatibles avec Recalbox !

Le résultat final

Et voilà, après 2 ans, beaucoup de temps et un peu d'investissement, une borne d'arcade terminée et prête à l'emploi. Ajouté à ceci un peu de domotique pour l'allumer automatiquement à la voix via un Google Home, et l'effet geek est réussi pour de bonnes soirées de retro gaming entre amis.

Pour faire le bilan de l'investissement, cela donne environ :

- Meuble : Bois, Plexi, T-Molding, visserie : 150 euros
- Peinture : 20 euros
- Ecran d'occasion : 30 euros
- Système audio : 54 euros
- Monnayeur : 70 euros
- Joysticks, boutons : 85 euros
- 2 manettes sans fils : 60 euros
- Système de jeux avec Raspberry : 55 euros

Soit un total d'un peu plus de 500 euros pour pouvoir s'amuser à plusieurs sur de bonnes parties et dire « je l'ai fait » (mais je le ne referai plus). •

Xavier Blanc
Pr. Institut Universitaire de France,
Université de Bordeaux
Expert Scientifique ProMyze

Coder *proprement*

Coder simplement ou coder proprement ?

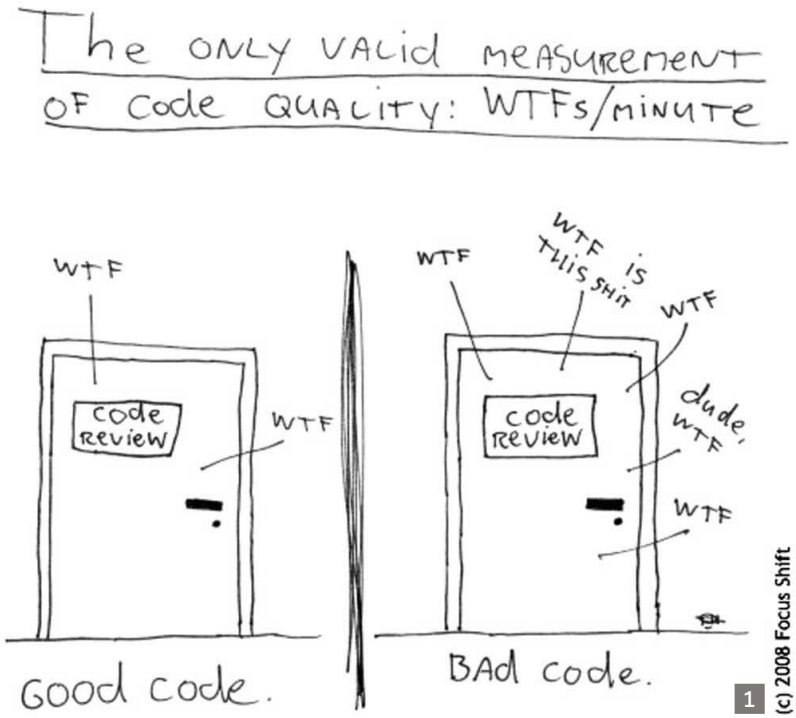
Programmer, coder, c'est ce que nous, développeurs, pratiquons tous les jours. Oui mais avec quel objectif ? Essentiellement, pour contenter le besoin formulé par l'utilisateur (ou le client). Celui qui utilisera le programme. Celui qui mesure difficilement la quantité d'efforts mis en œuvre pour réaliser ce programme. Celui qui, par contre, verra, voire même subira les malheureusement trop nombreux bugs que nous n'avons pas réussi à éviter. Ça c'est coder… simplement. C'est-à-dire coder jusqu'à ce que le programme marche ou jusqu'à ce que les tests passent. Est-il possible de faire mieux ? Certainement ! On doit pouvoir produire un code meilleur, un code de qualité, un code propre. Encore faut-il bien définir ce qu'est un code propre ? En effet, n'est-ce pas subjectif ? N'a-t-on pas tous en mémoire ce moment délicat où quelqu'un a regardé notre code et nous a dit brutalement : « c'est moche ! ». En caricaturant, on pourrait presque dire que la qualité d'un code et donc sa propreté se mesure au nombre de « c'est moche (traduction épurée de WTF) » qu'il reçoit lors d'une revue anonyme. 1

Mais ça, c'était avant ! En effet depuis de nombreuses années les développeurs discutent et débattent sur la propreté d'un code. Aujourd'hui, notamment grâce au livre « Clean Code » de Robert C. Martin, on peut affirmer qu'un code propre c'est avant tout un code lisible. L'idée principale du Clean Code est qu'un code lisible sera plus facile à comprendre, faire évoluer, débugger, améliorer, refactorer, optimiser, etc. En effet, tout code, même si sa principale vocation est d'être exécuté par une machine, va être lu et relu par d'autres développeurs. Il faut donc qu'il soit lisible.

Prenons un exemple pour s'en convaincre. Le code suivant vient de ExpressJS qui est le serveur web le plus utilisé dans la galaxie NodeJS. Ce code est dans le fichier express.js et vient au tout début, pour initialiser l'application. N'est-il pas difficile à lire ? La fonction createApplication() déclare une variable app, qui est une fonction, qui appelle app.handle() !

```
function createApplication() {
    var app = function(req, res, next) {
        app.handle(req, res, next);
    };
    //..
    return app;
}
```

Nous reviendrons sur ce code à la fin de cet article. Pour l'instant ce que l'on peut dire c'est qu'il marche mais qu'on doit pouvoir l'améliorer et le rendre plus propre. L'objectif étant de le rendre plus lisible pour faciliter la compréhension d'un développeur qui viendra rejoindre le projet.



1

Pourquoi coder proprement ?

Avant de préciser les principes qui composent le Clean Code et d'expliquer comment améliorer la qualité du code, il est important de bien préciser les motivations et les raisons qui doivent nous pousser à coder plus proprement.

La première raison est de faciliter la maintenance. Un code propre, plus lisible, sera plus facile à comprendre et donc plus facile à corriger. Les opérations de maintenance (correction de bugs ou ajout de fonctionnalité) seront donc plus faciles et, in fine, coûteront moins cher.

La deuxième raison est que le code propre, relu par plusieurs développeurs avant son intégration dans le projet, contient moins de bugs. En effet, un code propre sera plus facilement lu par vos collaborateurs et ceux-ci pourront alors plus facilement identifier les bugs qui s'y seront logés.

La troisième raison est qu'un code propre pousse les développeurs à coder proprement. Voire même il attire les développeurs et accroît leur motivation alors qu'un code qui n'est pas propre les repousse et les décourage. Ce troisième point fait l'unanimité chez les managers et les responsables des ressources humaines des sociétés d'informatique qui ont le vent en poupe. Ces sociétés savent que la motivation est l'élément clé de leur productivité. Elles savent aussi que les développeurs n'ont plus envie de participer à des projets mal codés et désertent les projets qui n'y prêtent pas une grande attention.

Enfin si nous analysons les raisons qui sont censées nous pousser à ne pas coder proprement (pas le temps, pas nécessaire, pas d'ajout fonctionnel, pas demandé par le management, trop mal codé par l'équipe d'avant pour pouvoir être amélioré, etc.), il faut se rendre à l'évidence, celles-ci sont des excuses pour masquer un manque de professionnalisme.

Les principes du Clean Code

Nous l'avons déjà dit, l'objectif du Clean Code est de rendre un code plus lisible. Le code que nous écrivons sera forcément lu, un jour, par un autre développeur, qui aura alors comme objectif de le modifier pour ajouter de nouvelles fonctionnalités ou de le corriger. Rendre un code lisible permet à cet autre développeur d'être plus efficace. De cet objectif général de lisibilité, un ensemble de principes a été recensé. Nous listons ici et illustrons par des exemples concrets les principes les plus importants. Ceux-ci s'appliquent à tous les langages de programmation. Notons que les principes présentés ici sont plutôt des guides que des règles. Libre à vous de les appliquer. Il ne faut pas qu'ils deviennent des contraintes. La seule règle du Clean Code étant la simplicité et le bon sens.

Le nommage des variables

Le nommage des variables est le premier ensemble de principes du Clean Code. En effet, tout code contient des variables. Il est donc très important de bien les nommer sinon il est impossible de savoir à quoi elles servent.

Trois principes pour nommer une variable :

- Une variable bien nommée est une variable dont on comprend le sens dès la lecture de son nom. Le nom doit porter l'intention ! Il faut donc tout faire pour éviter des noms qui ne portent pas de sens ou ceux dont l'intention est floue. A ce titre, l'affectation d'une valeur littérale à une variable lors de sa création (ex. customerLevel=5) s'appelle un magic number ou une magic value. Cela nuit à la lisibilité du code car il n'est pas possible de comprendre le sens de la valeur littérale.
- Pour faciliter le nommage des variables, il est préférable de choisir des noms prononçables et des noms cherchables (ctrl + F). Un nom prononçable sera plus facile à retenir et donc à comprendre. Une variable sera souvent cherchée dans le code. L'idéal est de tomber dessus en tapant les premières lettres de son nom dans la barre de recherche.
- Il existe deux dictionnaires pour trouver le nom d'une variable : le dictionnaire des mots de l'informatique (List, Set, Hash, State, Singleton, Factory, etc.) et le dictionnaire du métier ciblé par le programme. Un nom d'une variable devrait être composé de mots dans ces dictionnaires (ex. CustomerSet, ProductFactory, etc.).

Illustrons ces principes sur un exemple concret de code (j'ai choisi rabbitmq-java-client). La classe ConnectionFactory contient les variables (constantes) suivantes :

```
public static final String DEFAULT_USER = "guest";
public static final String DEFAULT_PASS = "guest";
```

Ces variables sont très bien nommées. On comprend l'intention. Par contre, la classe AMQConnection définit la variable suivante :

```
private final AMQChannel _channel0;
```

Cette variable pourrait être mieux nommée car le fait que ce channel soit le numéro 0 n'apporte pas d'information quant à son rôle. En lisant le code, on comprend que ce channel sert à l'initialisation et surtout à gérer la connexion. On pourrait améliorer ce code en le nommant « managementChannel ».

Enfin, toujours dans la classe ConnectionFactory, on peut voir le code suivant :

```
String userInfo = uri.getRawUserInfo();
if (userInfo != null) {
  String userPass[] = userInfo.split(":");
  if (userPass.length > 2) {
     throw new IllegalArgumentException("Bad user info in AMQP " +
        "URI: " + userInfo);
  }
}
```

Ce code utilise un magic number : 2 ! En lisant le code, on arrive à comprendre que l'objet est de vérifier qu'il y a bien un nom de login et un mot de passe dans userInfo et que ceux-ci sont séparés par le caractère ' :'. On pourrait améliorer ce code en l'isolant dans une petite fonction nommée containsUserAndPass(string s) qui définirait le magic number dans une constante.

Ces exemples sur le nommage des variables aident à comprendre les principes du Clean Code. On voit que les principes visent à améliorer la lisibilité du code. Ils ne sont pas à prendre comme des règles à suivre avec obligation mais plutôt comme des objectifs pour améliorer la lisibilité du code.

Les fonctions

La fonction est l'élément central d'un programme. Les fonctions servent à décomposer le code et donc le rendre plus lisible. Le nommage et la constitution des fonctions sont plus qu'importants pour faciliter la lisibilité d'un code.

Les 3 principes Clean Code pour les fonctions :

- Une fonction devrait concentrer une et une seule intention. Elle devrait rester simple, ne cibler qu'un seul traitement, à un seul niveau d'abstraction. Elle devrait être petite, avec peu de variables, peu de branches if, peu de return, etc. Une fonction est difficile à lire et à comprendre quand elle fait plusieurs traitements, que son contrôle est complexe, qu'elle a plusieurs variables, etc. L'idéal est donc de faire des fonctions simples.
- Une fonction devrait avoir peu de paramètres et ceux-ci devraient être exclusivement des paramètres d'entrée. Un ou deux paramètres c'est idéal. A partir de trois paramètres, il y a de fortes chances que tous les paramètres n'aient pas la même importance. De plus, la combinatoire des valeurs des paramètres devient très complexe à gérer et rend l'écriture des tests de la fonction quasiment impossible. Si les paramètres de la fonction sont des options, il est alors conseillé de mettre toutes les options dans un ensemble et de passer cet ensemble comme unique paramètre de la fonction. Enfin, lorsqu'on lit une fonction, on considère naturellement que les paramètres de la fonction sont utilisés en entrée et restent inchangés. Ainsi, une fonction ne devrait pas changer les paramètres qu'elle manipule.
- Une fonction devrait être nommée par un verbe, et les noms des paramètres devraient s'accorder avec ce verbe. Cela permet de comprendre très rapidement l'intention de la fonction. Enfin, si une fonction appelle d'autres fonctions, ces autres fonctions devraient être définies après dans le fichier. Il est plus facile de lire et de comprendre une fonction sans savoir ce que font les fonctions appelées dans le détail.

Illustrons ces trois principes sur notre exemple de rabbitmq-java-client.

La fonction setUri de la classe ConnectionFactory pourrait être améliorée. En effet, son nom laisse penser qu'elle n'est qu'un « setter », c'est-à-dire qu'elle affecte l'attribut uri de l'objet. Pour autant, cette fonction fait bien plus de choses. Elle décortique l'uri passée en paramètre et effectue plusieurs traitements. Son code n'est donc pas si simple car elle effectue plusieurs vérifications et plusieurs configurations. En particulier, elle met en place le protocol ssl (useSslProtocol) si l'uri passée en paramètre est une uri sécurisée. Cette fonction n'est donc pas un « setter » classique. On pourrait améliorer ce code en le décomposant en plusieurs fonctions, certaines s'occupant des vérifications et d'autres des configurations.

```
public void setUri(URI uri) throws URISyntaxException,
NoSuchAlgorithmException, KeyManagementException {
  if ("amqp".equals(uri.getScheme().toLowerCase())) {
     // nothing special to do
  } else if ("amqps".equals(uri.getScheme().toLowerCase())) {
     setPort(DEFAULT_AMQP_OVER_SSL_PORT);
     // SSL context factory not set yet, we use the default one
     if (this.sslContextFactory == null) {
        useSslProtocol();
     }
  } else {
```

La classe ConnectionFactory contient une multitude de fonctions newConnection permettant de construire une nouvelle connexion avec différents paramètres. Le code suivant présente un sous-ensemble de ces fonctions.

```
public Connection newConnection(Address[] addrs) {
...

public Connection newConnection(AddressResolver addressResolver) {
...

public Connection newConnection(Address[] addrs, String clientProvidedName)
 {
...

public Connection newConnection(List<Address> addrs) {
...

public Connection newConnection(List<Address> addrs, String clientProvidedName) {
...

public Connection newConnection(ExecutorService executor, Address[] addrs) {
...

public Connection newConnection(ExecutorService executor, Address[] addrs, String clientProvided
Name) {
...
```

On comprend alors que la multiplication des paramètres rend la lecture de cette classe plus difficile. Pire, la rédaction de la fonction qui réalise concrètement la création de la connexion est très complexe car elle doit gérer toutes les combinaisons possibles :

```
if (this.metricsCollector == null) {
  this.metricsCollector = new NoOpMetricsCollector();
}
// make sure we respect the provided thread factory
FrameHandlerFactory fhFactory = createFrameHandlerFactory();
ConnectionParams params = params(executor);
// set client-provided via a client property
if (clientProvidedName != null) {
  ...
  params.setClientProperties(properties);
}
if (isAutomaticRecoveryEnabled()) {
...
```

On voit bien que cette fonction réalise de très nombreux tests. Cela rend le développement bien trop complexe et source de nombreuses erreurs. Ces exemples montrent bien l'importance des principes Clean Code sur les fonctions.

Les commentaires

Les commentaires font partie intégrante du code, mais qu'en est-il de leur qualité ? Quelles parties du code faut-il commenter ? Et comment doit-on commenter son code ? Faut-il utiliser un langage particulier tel que JavaDoc qui permettra la génération de pages web ? Ou utiliser des commentaires textuels sans aucun marqueur ? A toutes ces questions, les adeptes du Clean Code répondent par une seule et même réponse : ***« commenter un mauvais code ne le rend pas propre, il vaut mieux améliorer son code plutôt que le commenter ! »***

Ce principe très tranché ne doit pas être résumé trop brutalement : Non, le code n'est pas son propre commentaire. Le code a besoin d'être commenté, mais pas tout le temps. Uniquement lorsque c'est absolument nécessaire. Principalement quand il n'est pas possible d'exprimer dans le code ce que le commentaire contient. Pour le reste, quand les commentaires ne servent pas la compréhension du code, ils sont inutiles et il vaut donc mieux les supprimer.

Voici donc les principes Clean Code pour les commentaires :

- Les aspects de licence du code doivent être exprimés dans des commentaires. Ces commentaires même s'ils prennent beaucoup de place au début des fichiers de code ne nuisent pas à la lecture du code. Ils sont absolument nécessaires. Leur présence n'est donc pas à débattre.
- Parfois préciser l'intention d'un morceau de code nécessite un haut niveau d'expressivité. Dans ces cas, le nommage des variables et des fonctions n'est pas assez expressif, on peut alors utiliser des commentaires.
- Les commentaires peuvent être utilisés pour expliquer les raisons d'une construction du code. Il est des cas où un bug d'une plateforme d'exécution nécessite une construction particulière (bizarre) du code, un commentaire permet ainsi d'indiquer les motivations de cette construction, et ainsi empêcher les futurs développeurs de vouloir la changer.
- Les commentaires doivent être utilisés pour documenter les API. Si une partie du code est une API et qu'il a pour objectif d'être utilisé par un autre code, il faut alors le commenter et, si possible, utiliser des langages spécifiquement construits pour commenter les API.

A l'inverse, voici les principes qui précisent quand il ne faut pas commenter :

- Il ne faut pas commenter le code quand le nommage est suffisamment précis. Il est des commentaires qui n'ajoutent aucune information. Pire, certains ajoutent à la confusion. Il faut alors supprimer ces commentaires, le code n'en sera que plus simple à lire.
- Il ne faut pas commenter le code pour exprimer les raisons ou les choix qui font ce qu'il est. Ces informations n'aident pas à la compréhension.
- Il ne faut pas expliquer dans le code les besoins utilisateurs. Il existe des outils bien plus adéquats pour les préciser. Ceux-ci ne doivent pas apparaître dans le code.
- Il ne faut pas utiliser les commentaires pour définir des parties du code ou pour séparer des morceaux du code. On voit parfois des commentaires utilisés pour marquer des places dans le code, pour préciser que telle ou telle partie du code sert telle ou telle intention. Ces commentaires montrent que le code est mal structuré et qu'il devrait être possible de l'améliorer en le repartitionnant en plusieurs fichiers par exemple. Ces commentaires devraient donc à terme disparaitre.
- Il ne faut pas utiliser le code pour commenter le code. En effet, lorsque des lignes de code sont commentées, les développeurs ont souvent peur de les supprimer et celles-ci restent là, commentées, rendant la lecture du code moins agréable.

Le code du projet rabbitmq-java-client regorge de perles sur les commentaires. Les auteurs de ce code ont trop commenté leur code et la plupart des commentaires n'aident absolument pas à la compréhension. Le morceau de code suivant contient deux beaux exemples. Le premier commentaire (« Divide by four… ») n'aide pas beaucoup et en plus il y a là un magic number. Il serait bien plus lisible de définir une constante (MAXIMUM_DELAY) et avec un simple commentaire motivant la valeur de celle-ci (l'intention), sans expliquer pourquoi on obtient cette valeur en multipliant par 250. Le deuxième commentaire (« should do more here ? ») est juste improbable. Il n'aide en rien la compréhension du code. Pourquoi récupérer une exception et ne pas la traiter ? Est-ce une faute dans le code ou une construction particulière qu'il ne faut pas changer ? 2

Il y a bien d'autres exemples similaires dans le code de cette application qui permettent de bien réaliser à quel point les commentaires qui n'aident pas la compréhension du code devraient être supprimés.

sticker

Ce commentaire ne montre-t-il pas un gros défaut dans le code ???

2

```
public void setHeartbeat(int heartbeat) {
    try {
        _heartbeatSender.setHeartbeat(heartbeat);
        _heartbeat = heartbeat;

        // Divide by four to make the maximum unwanted delay in
        // sending a timeout be less than a quarter of the
        // timeout setting.
        _frameHandler.setTimeout(heartbeat * 1000 / 4);
    } catch (SocketException se) {
        // should do more here?
    }
}
```

Le CleanCode et les Linters

Maintenant que nous avons présenté des principes du Clean Code, il nous est possible d'expliquer le lien avec les linters, vous savez ces outils tels que SonarQube, PMD, CheckStyle, ESLint, etc.

En effet, le Clean Code nous explique qu'il faut faire la chasse aux défauts dans le code. Un défaut, lorsqu'il est présent, rend le code plus difficile à lire. Nous avons vu par exemple qu'une variable mal nommée est un défaut, ainsi qu'un trop grand nombre de paramètres dans une fonction, ou la présence d'un commentaire inutile. Un défaut n'est pas la source de bug. Ce n'est pas parce qu'il y a un défaut dans votre code qu'il y a un bug. Pour autant, plus il y a de défauts dans votre code, plus il y a de chances qu'il y ait des bugs. En effet, si votre code est plein de défauts, sa lecture est vraiment complexe, et il y a de fortes chances qu'il contienne des bugs. Du coup, pour éviter de faire des bugs, le Clean Code préconise de faire le moins de défauts possible. Pour améliorer son code, il faut alors identifier les défauts qu'il contient et les supprimer. C'est là que les Linters interviennent.

Un Linter est un outil capable de détecter automatiquement des défauts dans le code. Il contient une base de règles qui correspondent à des défauts bien connus. Par exemple, tous les linters sont capables d'identifier les fonctions qui contiennent trop de paramètres et même les fonctions qui contiennent trop de lignes de code ou celles trop complexes (trop de if, de boucles, etc.).

On peut alors utiliser un Linter pour identifier tous les défauts que contient notre code et commencer à les supprimer en améliorant notre code. Pour autant, c'est beaucoup plus facile à dire qu'à faire car les Linters identifient bien souvent trop de défauts. Il faut alors configurer leur base de règles, les paramétrer, et se définir une méthodologie de réduction des défauts.

Cela est le début de l'apprentissage et de l'application du Clean Code…

En conclusion : ce qu'il faut retenir !

Nous venons de présenter les principes les plus importants du Clean Code (nommage des variables, simplicité des fonctions et commentaires utiles). Ces principes visent à rendre un code plus propre, c'est-à-dire plus lisible. Un code lisible contiendra moins de bugs et sera plus facile à maintenir. D'ailleurs, en revenant sur le code ExpressJS du début de l'article, app sert en fait de proxy et délègue tous les appels à handle. Au début de la fonction createApplication le proxy est vide, mais à la fin il est lié à handle. On doit pouvoir améliorer ce code pour le rendre plus lisible !

Ces principes sont des guides et non pas des contraintes. Libre à vous de les appliquer, souvent, tout le temps, parfois. L'important étant de toujours viser une amélioration de votre code en ayant en tête qu'il sera lu par un autre développeur.

Le Clean Code et ses principes portent une seule et même idée, celle du bon sens et de la simplicité. Avoir un code simple n'est pourtant pas si facile et demande des efforts. Ces efforts seront récompensés au centuple, que cela soit dans la qualité de l'application, dans l'engagement et la motivation.

Si vous voulez aller plus loin dans les principes du Clean Code, je vous invite à lire le livre de R. C. Martin, mais aussi et surtout, de pratiquer avec d'autres développeurs des revues de code ! Vous améliorerez ainsi considérablement vos compétences de développeurs. •

Paquet Judicaël
Coach Agile & Devops, Judicaël accompagne les entreprises en France et en Suisse comme Renault Digital ou la RATP dans leur transformation avec Myagile Partner qu'il a créé en janvier 2017.

MYAGILE PARTNER

Agilité & développeurs / DevOps : une bonne idée ?

Le métier de développeur est en train de changer depuis quelques années avec l'arrivée massive des méthodes agiles, de l'agilité et du mouvement Devops. Est-ce que cette transformation est une bonne idée ? Qu'impliquent ces nouvelles façons de travailler ? Doit-on réapprendre le métier de développeur ? Nous allons profiter de cet article pour répondre à l'ensemble des questions que vous pourriez vous poser sur le sujet.

Pour bien répondre à l'ensemble de ces questions, nous allons commencer par définir certains termes cités dès l'introduction de l'article et les expliquer de façon concise.

Qu'est-ce que l'agilité ?

L'agilité est une nouvelle culture d'organisation des entreprises qui a trouvé son nom en 1991 grâce à Nagel en popularisant le terme grâce au mouvement « The Agile Manufacturing Enterprise Forum ». Ce mouvement se base lui-même sur d'autres idées intéressantes comme le Design organisationnel (1977) et d'autres nouvelles visions d'organisations. L'agilité est une nouvelle culture d'entreprise et d'organisation qui commence à émerger ; amener une entreprise à vouloir devenir agile est une tâche complexe et longue.

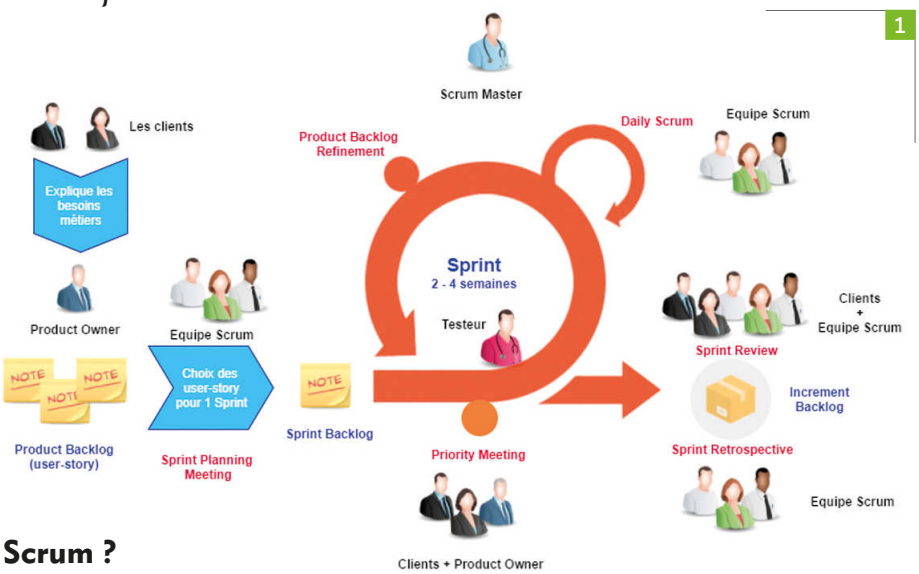
Qu'est-ce qu'une méthode agile ?

17 spécialistes se sont réunis pour réaliser le manifeste agile qui est une simple liste qui propose 4 valeurs fondamentales et 12 principes sous-jacents de ces valeurs.

Les méthodes agiles sont des méthodes de gestion de projet (qu'on préfère même appeler gestion de produit) qui fonctionnent autour des 4 valeurs suivantes :

- **Les individus et leurs interactions** plus que les processus et les outils ;
- **Des logiciels opérationnels** plus qu'une documentation exhaustive ;
- **La collaboration avec les clients** plus que la négociation contractuelle ;
- **L'adaptation au changement** plus que le suivi d'un plan.

Rappel du manifeste agile : "Nous reconnaissons la valeur des seconds éléments, mais privilégions les premiers."



1

Et le Scrum ?

En France, le Scrum est de loin la méthode agile la plus populaire au sein des DSI. Elle propose un cadre léger :

- des sprints (itérations courtes) et leurs évènements ;
- 3 rôles clés (le scrum master, le product owner, l'équipe de développeurs) ;
- un backlog avec des demandes.

Les équipes agiles dans les entreprises françaises qui font du Scrum ajoutent de plus en plus de technique d'ingénierie logicielle venant de l'Extreme Programming comme la TDD, la BDD voire du pair programming… L'excellence technique est devenue un des objectifs majeurs de nos entreprises car les conséquences de l'oubli total de qualité pour accélérer les livraisons ces quinze dernières années commencent à se faire ressentir : coût trop important de la maintenance, refonte obligatoire car l'évolution semble devenue impossible voire trop chère sur certains produits.

On peut même citer un phénomène récent mais qui s'amplifie avec le temps : la pénurie de développeurs qui s'amplifie année après année leur donne du pouvoir et augmentent leurs exigences. Les développeurs aujourd'hui refusent de rejoindre une entreprise pour travailler sur des applications trop vieillissantes… On peut donc rajouter un véritable problème de recrutement en plus du fait des coûts de maintenance parfois exorbitants. Le Scrum va définir 3 valeurs fondamentales à respecter : la transparence, l'inspection et l'adaptation que je ne détaillerai pas au sein de cet article. Comme le montre notre image présentant un Scrum de nouvelle génération, les équipes Scrum favoriseront énormément :

- le côté user-centric (le client sera à la base du produit, le client fera régulièrement des feedbacks pour incrémenter le produit d'évolution à chaque sprint) ;
- l'amélioration continue (rétrospective pour s'améliorer à chaque sprint) ;
- la livraison par petits blocs fonctionnels afin de tester chaque partie du produit rapidement. 1

Le développeur dans une équipe agile

Le développeur dans une équipe agile, voire une équipe Scrum n'est pas du tout le même qu'un développeur qui travaille dans un projet sous une méthodologie cycle en V et Waterfall. Vulgairement, on nommait ceux-ci des « pisseurs de code » mais cela a bien changé avec l'arrivée des méthodes agiles et du devops.

Aujourd'hui le développeur devient une denrée rare et les entreprises s'en rendent bien compte : des salaires en constantes évolution, des difficultés de trouver les profils idéaux, … En plus de cela, l'agilité amène les développeurs à avoir des rôles clés au sein des entreprises, ce qui n'était que trop rarement le cas dans le passé.

Développeur mais une équipe avant tout

L'individualisme n'existe plus dans le monde de l'agilité ! Nous parlons à présent d'équipe de développement.

L'agilité met fin au chef de projet technique qui veut tout gouverner au sein du projet ou au CTO autoritaire… L'équipe de développement dans le monde de l'agilité devient 100% responsable de l'aspect technique du produit et fait disparaitre peu à peu ces profils pour faire apparaître une véritable intelligence collective. Les managers qui deviennent souvent des Servant Leader, accompagneront les équipes de développement à être auto-organisée.

Comme vous l'aurez compris avec ces dernières lignes, travailler en équipe est un pilier de l'agilité : chaque succès sera un succès d'équipe et chaque échec sera un échec d'équipe. Il est vraiment important de bien comprendre que l'exploit individuel n'existe plus dans un univers Agile.

Dans les produits qui imposent par leur taille la mise en place d'une agilité à l'échelle (plusieurs équipes Scrum ou autres types d'agilités autour d'un même produit), la décision technique revient à l'ensemble des membres de la communauté technique. Cela permet de laisser le pouvoir de décision aux équipes de développement mais également de garder une cohérence technique générale entre l'ensemble des équipes de développement. C'est simple à dire mais la mise en place d'une communauté technique n'est pas une chose aisée.

Le développeur Agile est pluridisciplinaire

Dans une équipe agile, l'équipe de développeurs ne représente plus une simple équipes de « pisseurs de code »; chacun des programmeurs devient un membre pluridisciplinaire. Voici l'ensemble des rôles qu'on donne aux développeurs au sein des équipes agiles :

- ils font la documentation technique (qui était souvent réalisée par le chef de projet auparavant) ;
- ils réalisent les développements demandés ;
- ils sont responsables à 100% de la partie technique du produit… Ils prendront ensemble des décisions de façon collégiale sur les choix techniques à mettre en place ;
- Ils estiment en équipe et eux même la charge de travail ;
- Ils participent à l'affinage du Sprint backlog (soit des demandes à développer) ;
- Ils sont responsables de la qualité technique. Ils doivent impérativement tester le travail réalisé (en automatisant au maximum) avant de livrer les nouvelles demandes ;
- Ils doivent mettre en place une amélioration continue.

Comme on peut le voir ci-dessus, le développeur est devenu un acteur majeur des projets avec la mise en place des méthodes agiles ; il est devenu indispensable aux yeux de l'entreprise car on lui associe à 100% la stratégie technique des produits qu'on réalise. S'il y a 15 ans, on voyait le chef de projet comme l'indispensable des pôles IT, il a aujourd'hui laissé sa place à l'équipe de développement dans l'organisation de l'entreprise. Si on voit encore quelques chefs de projets, il faut avouer que ce type de poste semble vraiment en voie d'extinction.

Le développeur doit faire du KISS

Le travail d'équipe impose aussi de développer différemment en utilisant par exemple des concepts comme le KISS (acronyme de Keep it Simple, Stupid) pour développer.

Le principe KISS en agile est de faire le code le plus simple possible pour qu'il soit beaucoup plus facile à faire évoluer et à maintenir. Le but de ce principe est d'éviter à tout prix la sur-inflation fonctionnelle (feature creep) d'une application. Les conséquences ne sont pas anodines puisque la complexité du code coûte très cher à maintenir pour les entreprises voire impose parfois des refontes intégrales.

Faire simple insinue :

- d'utiliser des framework existants aux communautés actives pour ne pas réinventer la roue ;
- de ne pas créer des algorithmes inutilement complexes quand cela n'est pas nécessaire ;
- de bien documenter le code source ;
- de s'arrêter à faire les fonctionnalités demandées sans prévoir d'éventuelles demandes.

Parfois même au plus simple, le code peut être difficile à reprendre donc le développeur agile doit vraiment faire l'effort de simplifier à l'extrême la lecture (sans dénaturer la demande initiale).

Voici le type de situation que j'ai déjà rencontrée lors d'une de mes missions de coaching agile :

Les développeurs avaient perdu du temps sur les premières user-stories en tentant de mettre en place un système d'internationalisation car ils étaient persuadés que le site deviendrait multilingue ; cependant, cela n'est jamais arrivé, ce qui a complexifié inutilement le code (du code obsolète avant même d'être livré) et a fait perdre du temps aux développeurs pour développer des choses réellement utiles. C'est un classique que tout coach agile rencontre encore aujourd'hui.

Les développeurs ne doivent pas prévoir en amont des fonctionnalités potentielles et devront accepter de faire du refactoring au moment où les fonctionnalités seront demandées s'il y en a besoin même s'ils n'aiment pas procéder ainsi.

Qu'est-ce que le Devops ?

Si certains comprennent parfaitement ce que représente le devops, je vois encore de nombreuses entreprises utiliser le mot « devops » sans en comprendre réellement le sens. Si le mouvement du devops fait bien référence à l'automatisation des tests unitaires ou fonctionnels avec la mise en place de l'intégration continue ou à l'automatisation des livraisons, ce n'est pas l'aspect principal qu'évoque le mouvement Devops.

Le Devops est un mouvement qui privilégie la mise en place d'un alignement de l'ensemble de la DSI autour d'objectifs

communs ; le terme Devops est la concaténation de dev pour développeur et ops pour opérationnels, soit les ingénieurs responsables des infrastructures.

Avoir une équipe enfermée dans une pièce totalement isolée des équipes de développeurs pour mettre en place des solutions d'intégration continue ou de livraison continue ne correspond pas à ce concept Devops. C'est pourtant cette façon de faire que nous voyons de plus en plus aujourd'hui.

Qu'est-ce qu'un ingénieur Devops et un coach Devops ?

Il existe de plus en plus d'ingénieurs Devops qui viennent accompagner les équipes de développement pour mettre en place des stacks techniques complètes de nouvelles générations avec de l'intégration continue, de la livraison continue voire du déploiement continu ainsi que des technologies très récentes comme Docker, Kubernetes/Docker Swarm et des hébergements en Cloud.

On s'est cependant rapidement rendu compte que ce travail était souvent bien plus titanesque qu'imaginé avant de lancer certains chantiers, et qu'il était assez compliqué pour ces profils de partager leur temps entre accompagnement et mise en place de stacks techniques.

Des coachs agiles ayant une bonne maîtrise des pratiques d'ingénierie de qualité logicielle (anciens architectes et/ou Devops) viennent au sein des entreprises en tant que coachs Devops pour combler ce manque d'accompagnement obligatoire pour réussir une transformation Devops. Ce type de transformation est souvent sous-estimé alors qu'elle est assez difficile à réaliser.

Ces deux profils se complètent plutôt bien mais ils impliquent un investissement financier non négligeable ; les petites entreprises ne pourront pas mettre cette pratique en place bien qu'elles puissent en avoir besoin.

Comment le Devops fait évoluer le métier de développeur ?

Le mouvement Devops amène avec lui une nouvelle vision technique dans la création de produits. La mise en place de plateformes micro-services est une nouvelle démarche de développement qui peut rapidement s'avérer plus compliquée qu'elle ne peut le paraître en début de développement. Si on ajoute les nouvelles notions obligatoires qu'amènent les hébergements cloud, les développeurs seront dans l'obligation de connaître de nouvelles technologies comme Docker, API, techniques de tests comme la BDD et TDD… pour ne pas être totalement noyés.

Les développeurs doivent s'accaparer impérativement des connaissances les approchant de l'architecture si l'entreprise désire vraiment aller vers une démarche Devops. Cependant vouloir que tous les développeurs maîtrisent les technologies utilisées dans l'univers Devops reste utopique d'où la mise en place d'un ingénieur Devops au sein de ces équipes agiles.

2

Est-ce une bonne évolution ?

Les développeurs doivent évoluer pour suivre ces deux mouvements populaires qui se déploient très rapidement au sein de l'ensemble des DSI françaises.

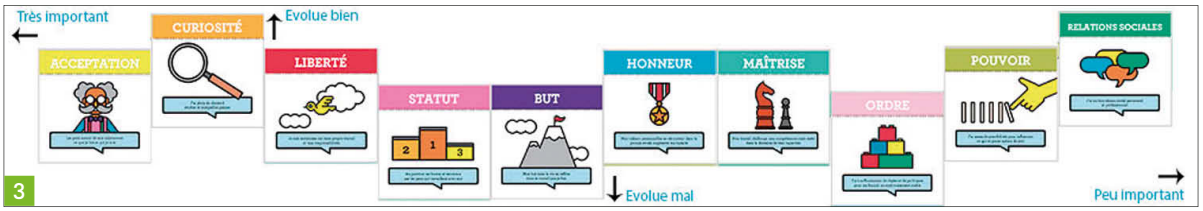
L'agilité et le devops sont de très bonnes évolutions tant elles apportent aux DSI et au produit final. Le fait d'axer le développement de produit sur des aspects user-centric permet d'avoir des produits beaucoup plus proches de ce qu'attendent les utilisateurs clés ; le Devops permet d'assurer un maximum de qualité technique dans les différentes livraisons réalisées.

Cependant, il faut faire très attention à ne pas vouloir tout changer trop vite et accepter que les choses prennent du temps. Si les développeurs sont de plus en plus nombreux à avoir eu de l'expérience dans un univers agile et Devops, ce type de transformation reste complexe à opérer.

Pour identifier les compétences à travailler voire à s'accaparer, il ne faut pas hésiter à mettre en place des exercices tels que la matrice de compétence. Elle permet d'identifier de façon simpliste les compétences attendues, les compétences présentes au sein de l'équipe et les compétences manquantes ; à partir de ces compétences manquantes, le management devra mettre en place des solutions pour se les accaparer :

- des BBL (Brown Bag Lunch) pour avoir une petite formation d'1h en échange d'un simple repas le midi ;
- des formations par des instituts reconnus pour réaliser des formations de qualité sur plusieurs jours ;
- des formations internes aux équipes si certains membres maîtrisent une compétence clé exigée pour chacun des membres de l'équipe.

Regardez un exemple de matrice de compétences d'une équipe de développement agile (image 1) qui doit également avoir des compétences Devops pour être performante ; cette équipe est également accompagnée d'un Scrum Master et d'un Product Owner. On voit sur cet exemple que l'équipe actuelle n'a pas l'ensemble des compétences exigées pour mener a bien le projet (lignes entourées en rouge). Les managers vont donc mettre des solutions en place avec l'aide de cette matrice de compétence :



- proposer à Eric de former certains développeurs pour qu'ils soient capable d'intervenir sur ReacJs ; sachant que l'idée n'est pas d'avoir deux personnes qui maîtrisent parfaitement à 100% la compétence, ce travail devrait être relativement rapide ;
- il faudra impérativement former l'un des membres de l'équipe à RabbitMQ ; cependant, la maîtrise étant exigée, la matrice de compétence montrerait plutôt le besoin de recruter un développeur spécialiste sur cet outil ;
- excellente nouvelle, l'équipe est constituée d'au moins deux développeurs ayant de très bonnes connaissances en agile ; le Scrum Master formera le reste de l'équipe pour que tous les développeurs maîtrisent parfaitement leur rôle au sein de ce nouveau contexte agile ;
- pour travailler sur des environnements cloud de nouvelle génération, il est indispensable que l'ensemble de l'équipe soit capable d'intervenir sur l'outil Docker. Le manager demandera à John et Robert de former le reste de l'équipe pour que chaque membre ait les connaissances nécessaires pour intervenir sur Docker. 2

Quand ces évolutions sont bien définies, il est également fortement conseillé de suivre l'évolution de la motivation de chaque membre de l'équipe ; car si chaque membre peut s'accaparer de nouvelles compétences, voire de développer différemment par rapport à avant, cela ne veut pas dire que ce nouveau contexte les motive. L'exercice du Moving Motivators venant tout droit du management 3.0 créé par Jurgen Appelo permet de connaître les motivations réelles des personnes réalisant l'atelier.

Le Scrum Master proposera à chaque développeur de l'accompagner dans un lieu relativement calme avec une table où il pourra disposer les cartes.

Voici la liste des cartes présentes : **acceptation, curiosité, liberté, statut, but, honneur, maîtrise, ordre, pouvoir, relations sociales**.

Le Scrum Master commencera par demander au développeur de placer de gauche à droite les cartes de la plus importante à la moins importante qu'il accorde à chaque carte de moving motivators. Le développeur pourra prendre une dizaine de minutes pour réfléchir à cela.

Le Scrum Master proposera ensuite au développeur sur une deuxième phase de 15 minutes de pousser les cartes si la transformation de l'entreprise a un impact positif sur l'une des motivations et de tirer vers lui les cartes si la transformation de l'entreprise a un impact négatif sur lui. Si la transformation a selon lui aucun impact, il ne bouge pas la carte (3).

Quand il y a plus de carte en "Evolue mal" qu'en "Evolue bien", il faut sérieusement envisager de voir avec le développeur si la transformation n'est pas un risque de départ.

Une très bonne évolution mais à bien gérer

En effet, l'évolution du métier de développeur avec l'arrivée massive de l'agilité et du Devops dans les entreprises françaises est une très bonne chose, mais il est indispensable d'accompagner les équipes qui manquent souvent encore de maturité pour se lancer seules dans cette aventure. Sans cet accompagnement, l'expérience peut se transformer en véritable cauchemar par l'entreprise. •

Aurélie Vache
Développeuse chez Continental Intelligent Transportation Systems
Duchess France & DevFest Toulouse Leader
Toulouse Data Science core-team & Marraine Elles bougent
@aurelievache

Infrastructure as Code avec Terraform

Vous ne voulez plus créer manuellement vos machines physiques, vos VMs, vos VPC, vos conteneurs, vos lambda … ? Nous allons voir dans cet article qu'il existe un outil permettant de faire de l'Infrastructure as Code qui vous permettra de passer du ClickOps au DevOps.

Nous commencerons par voir ce qu'est Terraform, les concepts, sa CLI. Nous ferons une mise en pratique de l'outil, puis nous verrons quelques tips et outils sympa. Nous ferons aussi une comparaison avec CloudFormation.

Qu'est ce que Terraform ?

Terraform est un outil créé en 2014 par HashiCorp, la société qui a créé d'autres outils que vous connaissez sûrement : Consul, Vagrant, Vault, Atlas, Packer et Nomad.

C'est un outil Open Source, écrit en Go, avec une communauté active de plus de 1000 contributeurs qui repose sur une architecture basée sur les plugins.

Terraform (TF) est un outil qui permet de **construire**, **modifier** et **versionner** une infrastructure. Contrairement à ce que l'on peut lire sur internet, la technologie n'est pas plateforme agnostic MAIS elle permet d'utiliser plusieurs providers dans un même template de configuration. Il existe en effet des plugins pour des providers de Cloud, des services d'hébergement, des SCM … Nous le verrons un peu plus tard dans cet article.

Que fait l'outil ?

- Il assure la création et la cohérence d'infrastructure ;
- Il permet d'appliquer des modifications incrémentales ;
- On peut détruire des ressources si besoin ;
- On peut prévisualiser les modifications avant de les appliquer.

HCL

Les fichiers de configurations s'écrivent en HCL (HashiCorp Configuration Language). Le principe est d'écrire des ressources.

Les ressources peuvent être écrites en JSON également mais il est recommandé de les écrire en HCL.

Lire un fichier de configuration HCL est plutôt simple et intuitif. 1

C'est un langage dit "human readable". Ce qu'il faut savoir c'est que Terraform scanne tous les fichiers se terminant par .tf dans le répertoire courant; en revanche, il ne va pas scanner les répertoires enfants.

CONCEPTS

On va voir ensemble quelques concepts :

Provider

Un provider est responsable du cycle de vie/du CRUD d'une ressource : sa création, sa lecture, sa mise à jour et sa suppression.

Plusieurs providers sont actuellement supportés :

- AWS, GCP, Azure, OpenStack …
- Heroku, OVH, 1&1 …
- Consul, Chef, Vault …
- Docker, Kubernetes …
- GitLab, BitBucket, GitHub …
- MySQL, PostgreSQL …
- mais encore RabbitMQ, DNSimple, CloudFlare …

La liste complète des providers est disponible sur le site de TF : https://www.terraform.io/docs/providers/index.html

Vous ne trouvez pas le provider de votre rêve ? Pas de souci vous pouvez écrire votre propre plugin et participer à l'élaboration de nouvelles fonctionnalités de Terraform. Si vous ne codez pas encore en Go, ce sera l'occasion de vous y mettre ;-).

Pour configurer un provider, nous voulons par exemple déployer une infra AWS, il suffit de créer une ressource "aws" dans un fichier se terminant par .tf :

root.tf :

```
provider "aws" {
  region = "eu-central-1"
}
```

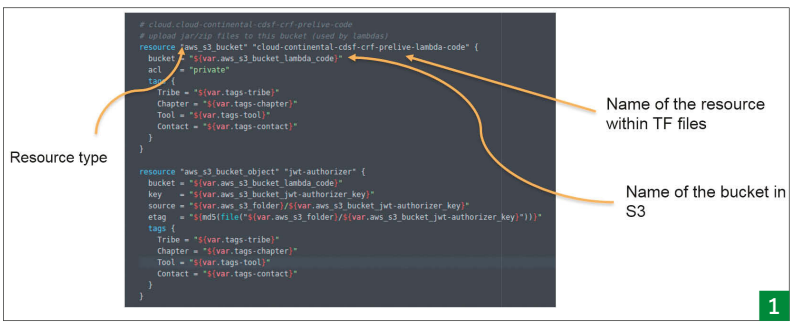
Bonne pratique : ne pas mettre les credentials directement dans la ressource provider aws mais configurez vos variables d'environnement :

```
$ export AWS_ACCESS_KEY=YOUR_ACCESS_KEY
$ export AWS_SECRET_ACCESS_KEY=YOUR_SECRET_KEY
$ export AWS_DEFAULT_REGION=eu-central-1
```

Ou bien vous pouvez les mettre dans un fichier .aws/credentials :

```
[default]
aws_access_key_id = YOUR_ACCESS_KEY
aws_secret_access_key = YOUR_SECRET_KEY
```

Attention, il va de soi que vous devrez prendre les tokens d'un utilisateur qui aura les droits/la bonne policy pour créer les services que vous voulez.



1

Variables

Afin de faire du code propre et qui se réutilise, il est recommandé d'initialiser des variables et de les utiliser dans les autres fichiers .tf.
Définissez vos variables dans un fichier variables.tf, par exemple :

```
variable "default-aws-region" {
 default = "eu-central-1"
}

variable "aws_account_id" {
 default = "123456789123"
}

variable "tags-tool" {
 default = "Terraform"
}

variable "tags-contact" {
 default = "Aurelie Vache"
}

variable "aws_s3_folder" {
 default = "s3"
}

variable "aws_s3_bucket_terraform" {
 default = "com.programmez.terraform"
}
```

Appelez-les dans vos ressources :

```
resource "aws_s3_bucket" "com-programmez-terraform" {
 bucket = "${var.aws_s3_bucket_terraform}"
 acl    = "private"

 tags {
  Tool    = "${var.tags-tool}"
  Contact = "${var.tags-contact}"
 }
}
```

Modules

Les modules sont utilisés pour créer des composants réutilisables, améliorer l'organisation et traiter les éléments de l'infrastructure comme une boîte noire.
C'est un groupe de ressources qui prennent en entrée des paramètres et retournent en sortie des outputs.
Dans le même fichier, root.tf, dans lequel vous avez défini votre provider, vous pouvez ensuite définir vos modules :

```
module "lambda" {
 source         = "./lambda/"
 toto           = "${var.aws_s3_bucket_toto_lambda_key}"
 titi           = "${var.aws_s3_bucket_titi_key}"
 tutu           = "${var.aws_s3_bucket_tutu_key}"
}
```

Outputs

Les modules peuvent produire des outputs que l'on pourra utiliser dans d'autres ressources.

lambda/outputs.tf :

```
output "authorizer_uri" {
 value = "${aws_lambda_function.lambda_toto.invoke_arn}"
}
```

aws_api_gw.tf :

```
resource "aws_api_gateway_authorizer" "custom_authorizer" {
 name                          = "CustomAuthorizer"
 rest_api_id                   = "${aws_api_gateway_rest_api.toto_api.id}"
 authorizer_uri                = "${module.lambda.authorizer_uri}"
 identity_validation_expression = "Bearer .*"
 authorizer_result_ttl_in_seconds = "30"
}
```

State

Un state est un snapshot de votre infrastructure depuis la dernière fois que vous avez exécuté la commande **terraform apply.** 2
Terraform utilise un stockage local pour créer les plans et effectuer les changements sur votre infra. Mais il est possible de stocker ce state, dans le cloud.
Pour configurer le stockage de ce state en remote, il suffit de définir un backend.

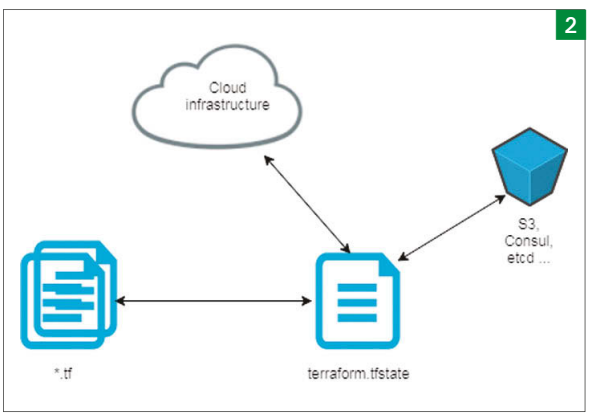
backend.tf :

```
# Backend configuration is loaded early so we can't use variables

terraform {
 backend "s3" {
  region  = "eu-central-1"
  bucket  = "com.programmez.terraform"
  key     = "state.tfstate"
  encrypt = true
 }
}
```

CLI

Terraform a une CLI (Command-Line Interface) facile à utiliser composée de plusieurs commandes, nous allons en voir quelques-unes, pas toutes, uniquement celles que j'utilise le plus au quotidien.

2



Init

```
$ terraform init
```

Cette commande va initialiser votre répertoire de travail qui contient vos fichiers de configuration en .tf.
C'est la première commande à exécuter pour une nouvelle configuration, ou après avoir fait un checkout d'une configuration existante depuis votre repo git par exemple.
La commande init va :

- télécharger et installer les providers ;
- initialiser le backend (si défini) ;
- télécharger et installer les modules (si définis).

Plan

```
$ terraform plan
```

La commande plan permet de créer un plan d'exécution. Terraform va déterminer quelles actions il doit faire afin d'avoir les ressources listées dans les fichiers de configuration par rapport à ce qui est actuellement en place sur l'environnement/le provider cible.
Cette commande n'effectue concrètement rien sur votre infra.
Bonne pratique : afin de sauver le plan, vous pouvez spécifier un fichier de sortie :

```
$ terraform plan -out programmez.out
```

Apply

```
$ terraform apply programmez.out
```

La commande apply, comme son nom l'indique, permet d'appliquer les changements à effectuer sur l'infra. C'est cette commande qui va créer nos ressources.
Attention, depuis la version 0.11 de Terraform, sortie le 16 novembre dernier, lorsque vous utilisez TF dans un environnement interactif, en local par exemple, mais pas en CI/CD, il est recommandé de ne plus passer par un plan mais de directement faire un apply et de répondre Yes si vous souhaitez appliquer ce plan.

Destroy

```
$ terraform destroy
```

La commande destroy permet de supprimer TOUTES les ressources. Un plan de suppression peut être généré au préalable :

```
$ terraform plan -destroy
```

Console

Grâce à la CLI console, vous pouvez connaître la valeur d'une ressource Terraform. Cette commande est pratique pour faire du debug, avant de créer un plan ou de l'appliquer.

```
$ echo "aws_iam_user.programmez.arn" | terraform console
arn:aws:iam::123456789123:user/programmez
```

Get

```
$ terraform get
```

Cette commande est utile si par le passé vous avez déjà fait un terraform init puis ajouté un module, il faut préciser maintenant à Terraform qu'il faut récupérer le module ou bien le mettre à jour. Si vous ne le faites pas, lors de d'un Terraform plan, TF vous demandera de le faire ;-).

Graph

```
$ terraform graph | dot -Tpng > graph.png
```

La CLI graph permet de dessiner un graphique de dépendances visuelles des ressources Terraform en fonction des fichiers de configuration. 3
Au bout d'un certain nombre de ressources dans un répertoire, Terraform n'arrive plus à générer ce graphique. J'espère que ce problème sera corrigé dans les futures version ;-).

TRÈVE DE BLABLA, PASSONS À LA PRATIQUE !

Commençons tout d'abord par installer Terraform :

```
$ curl -O https://releases.hashicorp.com/terraform/0.11.1/terraform_0.11.1_
linux_amd64.zip
$ sudo unzip terraform_0.11.1_linux_amd64.zip -d /usr/local/bin/
$ rm terraform_0.11.1_linux_amd64.zip
```

0.11.1 étant la dernière version de Terraform, stable, à ce jour.
Ces commandes vont extraire un binaire dans /usr/local/bin/, qui est déjà dans votre PATH.
Afin de vérifier que Terraform est correctement installé, veuillez vérifier la version courante de l'outil :

```
$ terraform --version
Terraform v0.11.1
```

Info: terraform -version fonctionne également.

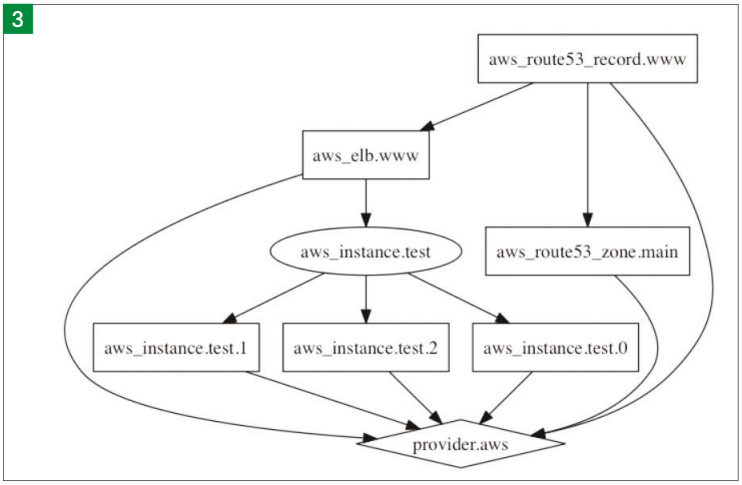
1/ Création d'un répertoire

```
$ mkdir programmez/
```

2/ Création d'un fichier .tf

```
$ vi aws_s3.tf
```

3



```
############## AWS S3 ##############
resource "aws_s3_bucket" "com-programmez-terraform" {
bucket = "${var.aws_s3_bucket_terraform}"
acl = "private"
  tags {
    Tool = "${var.tags-tool}"
    Contact = "${var.tags-contact}"
  }
}
```

3/ Pré-requis

Pour indiquer à Terraform sur quel compte AWS vous souhaitez déployer l'infrastructure souhaitée, vous devez définir des variables d'environnement AWS au préalable, par exemple dans un fichier .aws/credentials ou avec des variables d'environnement :

```
$ export AWS_ACCESS_KEY_ID="an_aws_access_key"
$ export AWS_SECRET_ACCESS_KEY="a_aws_secret_key"
$ export AWS_DEFAULT_REGION="a-region"
```

4/ Initialiser Terraform

```
$ terraform init
[0m[1mDownloading modules...[0m
[0m[1mInitializing the backend...[0m
[0m[1mInitializing provider plugins...[0m
The following providers do not have any version constraints in configuration,
so the latest version was installed.
To prevent automatic upgrades to new major versions that may contain breaking
changes, it is recommended to add version = "..." constraints to the
corresponding provider blocks in configuration, with the constraint strings
suggested below.
* provider.aws: version = "~> 1.6"
```

5/ Plan

```
$ terraform plan -out crf.out
Refreshing Terraform state in-memory prior to plan...
The refreshed state will be used to calculate this plan, but will not be
persisted to local or remote state storage.
aws_s3_bucket.com-programmez-terraform: Refreshing state... (ID: com.
programmez.terraform)
...
Plan: 1 to add, 0 to change, 0 to destroy.
```

6/ On applique le plan

```
$ terraform apply plan.out
...
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
```

OUTILS & TIPS

Terraforming

Afin de nous faciliter la vie, il y a quelques petits outils intéressants à connaître et à utiliser. Si vous avez une infra AWS existante et que vous devez la dupliquer dans plusieurs autres comptes, **Terraforming** (https://github.com/dtan4/terraforming) est fait pour vous. C'est un outil écrit en Ruby qui permet d'extraire des ressources AWS existantes et de générer un fichier Terraform correspondant.

Pour l'installer, voici la marche à suivre :

```
$ sudo apt install ruby
$ gem install terraforming
```

Pour extraire un type de ressources c'est très simple. Par exemple si l'on veut extraire des buckets s3 :

```
$ terraforming s3 > aws_s3.tf
```

Comme vous pouvez le constater si vous testez cette commande sur un compte AWS qui contient des buckets s3, cette dernière n'extrait que les buckets, elle n'extrait pas les objets : les zip, jar et divers fichiers contenus dans vos buckets. Il vous faudra du coup écrire les ressources `aws_s3_object` comme celle-ci :

```
resource "aws_s3_bucket_object" "authorizer_keystore" {
 bucket = "${var.aws_s3_bucket_authorizer}"
 key    = "${var.aws_s3_bucket_authorizer_keystore_key}"
 source = "${var.aws_s3_folder}/${var.aws_s3_bucket_authorizer_keystore_key}"
 etag   = "${md5(file("${var.aws_s3_folder}/${var.aws_s3_bucket_authorizer
_keystore_key}"))}"

 tags {
  Tool    = "${var.tags-tool}"
  Contact = "${var.tags-contact}"
 }
}
```

Attention, terraforming ne prend pas en compte toutes les ressources AWS, notamment les API Gateway. De ce fait, même si vous souhaitez dupliquer une API GW existante, il vous faudra vous les écrire à la mimine.

Gitignore

.gitignore :

```
#  Local .terraform directories
**/.terraform/*

# .tfstate files
*.tfstate
*.tfstate.*

# .tfvars files
*.tfvars

# .out files
*.out
```

Comment convertir un local state en remote state ?

Si vous avez déjà, par le passé, créé un local state et que vous avez maintenant défini un state en remote dans un fichier nommé backend.tf, par exemple, voici la marche à suivre pour uploader votre local state terraform.tfstate dans le cloud :

```
$ terraform init
$ terraform state push
```

Et CloudFormation, on l'oublie ?

Comme vous l'avez vu, nous avons opté pour Terraform comme

outil pour faire de l'IaC (Infrastructure as Code) pour nos services managés sur AWS mais nous aurions pu partir sur du CloudFormation. On ne s'outille pas pour suivre une mode, ou ce que fait le voisin mais pour répondre aux besoins d'un projet/d'un contexte donné.

Terraform

- Support de presque tous les services AWS et d'autres (cloud) providers ;
- Open Source ;
- Pas de rolling update pour les ASG (Auto Scaling Group) ;
- HCL/JSON ;
- State management ;
- Support de Vault.

CloudFormation

- Support de presque tous les services AWS (uniquement ce cloud provider!) ;
- Service géré et "offert" par AWS ;
- Le rolling update d'EC2 par un ASG est supporté ;
- JSON ;
- Pas de State management.

A noter qu'il est possible de charger du CloudFormation (CF) dans du Terraform, si par exemple vous avez déjà votre stack écrite en CF :

```
# Setup of an CloudformationStack with terraform
resource "aws_cloudformation_stack" "my_stack" {
  depends_on = [
    "aws_s3_bucket_object.lambda_code",
    "aws_s3_bucket_object.authorizer"
  ]
  name = "${var.cf_stack_name}"
  template_url = "https://${aws_s3_bucket.deployment_bucket.bucket
_domain_name}/cf_programmez.json"
  provider = "aws.region"

  parameters {
    LambdaCodeAuthorizer = "${basename(var.authorizer_lambda_jar)}",
    S3CodeBucket = "${aws_s3_bucket.deployment_bucket.id}",
    ...
  }

  tags {
    Tool   = "${var.tags-tool}"
    Contact = "${var.tags-contact}"
  }
}
```

AVANTAGES / INCONVÉNIENTS

Terraform n'est pas une baguette magique, cette techno a des inconvénients, comme pour toutes les technos. Il s'agit notamment d'un outil assez jeune, et en constante évolution donc il y a des bugs et toutes les ressources de tous les providers ne sont pas encore prises en compte.
On peut dresser une petite liste de "Pros" et de "Cons :

Avantages

- Permet de définir de l'Infrastructure as Code ;
- Syntaxe concise et lisible ;
- Réutilisation de : variables, inputs, outputs, modules ;
- Commande **plan ;**
- Multi cloud support ;
- Développement très actif.

Inconvénients

- Outil jeune (comporte des bugs) ;
- Pas de rollback possible ;
- Verbeux.

J'ajouterai que les messages d'erreurs ne sont pas super explicites :

```
* module.lambda.aws_lambda_function.lambda_toto: 1 error(s) occurred:

* module.lambda.aws_lambda_function.lambda_toto: aws_lambda_function
.lambda_toto: InvalidSignatureException: Signature expired: 20171218T075729Z
 is now earlier than 20171218T080647Z (20171218T081147Z - 5 min.)
  status code: 403, request id: 17c82bc7-e3cb-11e7-8a7e-6b18fbf66fae
```

CONCLUSION

Ma conclusion personnelle est que TF est un des outils qui permet de faire de l'Infrastructure as Code et de passer du ClickOps au DevOps. Mais faire du Terraform en local ce n'est pas la vraie vie, donc nous pourrons voir dans un prochain épisode comment automatiser la création de ressources écrites en Terraform dans une chaîne de Continuous Integration/Continuous Deployment avec notre fidèle compagnon : Jenkins. •

• Sacha Labourey,
CEO CloudBees

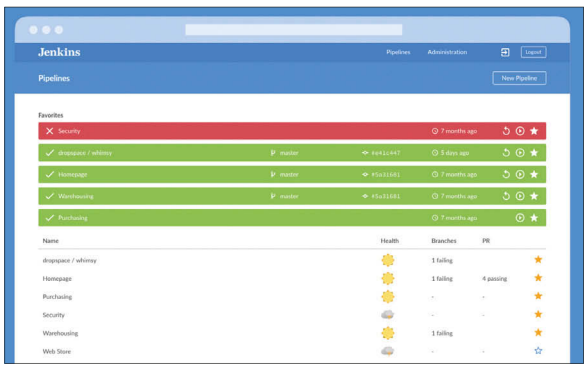
Cinq conseils pour développer ses pipelines de deploiement continu avec Jenkins Pipeline

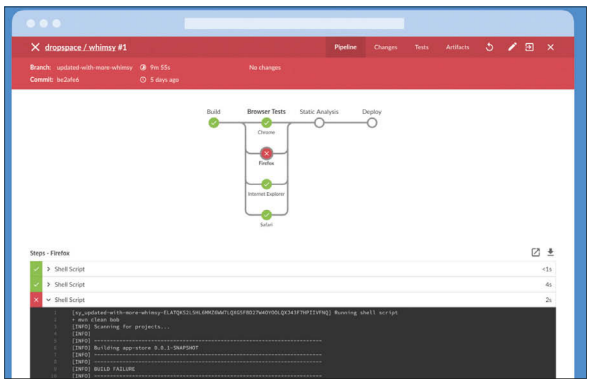
Lorsque des équipes IT décident de mettre en place des pipelines de déploiement continu, elles ont toutes les mêmes objectifs en tête. Elles veulent rationaliser leurs processus pour développer de meilleurs logiciels et créer une série d'étapes en cascade qui leur permettront d'aller d'un bout à l'autre du processus, le tout avec un minimum d'interventions humaines.

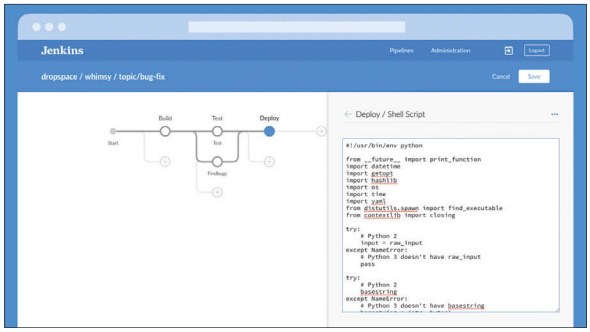
Mais comment y arriver ? Et quels sont les pièges types à éviter ? En suivant ces quelques conseils, les entreprises s'assurent de tirer le meilleur parti de leurs pipelines.

Modélisez votre chaîne de valeur existante

C'est la première étape pour faire passer vos processus d'un niveau débutant à un niveau avancé. Une fois votre pipeline défini, vous devez créer un modèle bien plus détaillé pour suivre le parcours de votre code à travers les différentes étapes du processus. Déterminez les étapes où des goulets d'étranglements peuvent se produire, celles où vous pourriez avoir à rebrousser chemin et celles où il est sans doute nécessaire d'effectuer des vérifications pour voir si la qualité est toujours au rendez-vous. Exécutez le pipeline d'un bout à l'autre du processus, du développement jusqu'aux phases d'assurance qualité, de pré-production et de production. Pour reprendre l'expression de Jez Humble, Directeur Technique chez DORA (DevOps Research & Assessment), : "modélisez la chaîne de valeur" ! Une manière de procéder pour modéliser cette chaîne de valeur consiste à vous appuyer sur un autre projet de votre organisation qui a des caractéristiques similaires au vôtre. Vous pouvez également commencer par le strict minimum : une phase de démarrage pour construire votre application et exécuter des métriques et des tests unitaires de base, une phase d'exécution des tests d'acceptance et une troisième phase de déploiement de votre application dans un environnement de type production afin que vous puissiez en faire la démonstration.







Mesurez les performances de votre pipeline

Maintenant que vous avez façonné votre pipeline, il faut tout mesurer. Analysez chaque étape et repérez où se situent les goulets d'étranglement. Puisque vous avez modélisé ces étapes, vous pouvez générer des analyses après chacune d'entre elles. L'objectif principal de cette étape est de faire autant de tests que possible avec un délai d'exécution minimal. Si vous n'utilisez pas déjà un processus de test automatique, il faut y remédier et vous créer un environnement pour réaliser des tests automatisés. Cela vous permettra de gagner du temps et de réduire le nombre d'incidents.

En procédant ainsi, Allianz Insurance a constaté "une réduction des incidents de priorité 1 et 2", affirme Adam Rates, Responsable de la stratégie et de l'architecture chez l'assureur.

Pour Sebastian Bosse, Chef de projet Intégration Continue chez Bosch, l'automatisation leur a permis de réaliser ce processus en 3 heures au lieu de 3 jours. "Nous avions décidé de nous lancer en prenant un projet vaste et complexe comme point de référence. En raison des dépendances et de facteurs externes qui ont souvent interrompu les heures de procédures manuelles requises, la construction de ce projet avait alors pris jusqu'à trois jours ", explique Sebastian Bosse. "Après avoir automatisé le pipeline de construction avec CloudBees, il se termine maintenant en deux heures et 20 minutes, ce qui se traduit par une augmentation de plus de 95% de la vitesse d'exécution."

Avant de vous lancer dans les mesures, demandez-vous quels sont les facteurs à prendre en compte pour les effectuer. Dans la plupart des cas, il s'agit de mesurer des éléments comme la rapidité de développement, le déploiement ou encore le retour clients, la fréquence des déploiements et des défaillances, le temps de réparation, le volume des demandes de réparation et le taux de changement de ces indicateurs.

Les mesures de DevOps portent généralement sur le déploiement, les opérations et la maintenance (par opposition à la phase initiale de conception et de développement, par exemple).

Apprenez vite de vos erreurs

Intuitivement, on pourrait penser que le flux d'un pipeline reste le même du début à la fin, mais ce n'est pas le cas. Bien sûr, vous voulez réaliser toutes les étapes aussi effi-

cacement et rapidement que possible, mais vous devez aussi ménager plus de temps pour les tests et prévoir plus de dispositifs pour relancer le processus à un stade précoce, afin de vous assurer du succès de chaque étape.

C'est par exemple ce qu'a fait la société Bosch lorsqu'elle a fait appel à CloudBees pour accélérer son développement logiciel. Grâce à Jenkins Pipeline, les développeurs de chez Bosch ont été capables d'orchestrer et de gérer des pipelines de livraison de logiciels complexes et d'évaluer la performance de chacune des étapes à l'intérieur de ces mêmes pipelines.

En résumé, si vous devez casser quelque chose, faites-le le plus tôt possible, faute de quoi les phases finales du pipeline demanderont des opérations plus lourdes, multipliant les possibilités d'échecs. Si un développeur applique une modification, il doit savoir s'il a fait une bêtise dans les cinq minutes, puis passer à l'étape suivante. Des tests plus lourds peuvent être exécutés toutes les 15 minutes par exemple. Dans certains cas, vous pouvez même lancer les tests en parallèle de votre pipeline. Les tests peuvent être effectués à intervalles réguliers à mesure que votre version évolue dans le pipeline.

Évitez les longs chemins critiques

Si vous créez un chemin trop long, des travaux peuvent se retrouver bloqués à cause de processus dont ils ne dépendent même pas directement. Si vous pouvez fractionner votre travail en morceaux, le processus se terminera plus rapidement. De plus, si vos tests automatisés prennent plusieurs jours, c'est bien trop long. Vous devez trouver du temps pour incorporer des feedbacks.

Il est important de noter que tout retard sur le chemin critique affecte in fine le délai de mise sur le marché de l'application. Or, en répondant aux exigences du chemin critique et en connaissant le temps qu'il requiert, il est alors possible de prendre des mesures pour le réduire grâce à l'automatisation et à d'autres techniques.

En termes de gestion de projet, cela peut également signifier faire en sorte que les individus ou les équipes les moins à l'aise avec l'exercice ne se voient pas attribués au "chemin critique". Pour tenter de réduire les risques, il peut également s'avérer utile de rédiger et rendre disponibles des documents énumérant tous les éléments susceptibles de mal tourner et d'indiquer comment y remédier pour remettre le projet sur les rails.

Trouvez des moyens de lancer des opérations en parallèle

Un bon moyen de gagner du temps et d'économiser de l'énergie est d'exécuter des travaux en parallèle. Pourquoi perdre du temps quand des machines peuvent faire le travail en arrière-plan ?

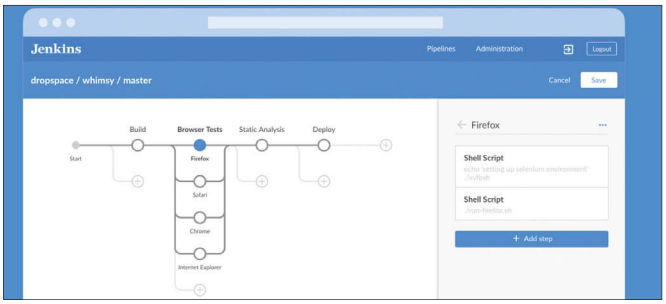
L'exécution des étapes en parallèle permet de diviser les tests ou de tester le code en fonction de différents environnements ou versions linguistiques. Il est ainsi possible d'obtenir un feedback plus rapide.

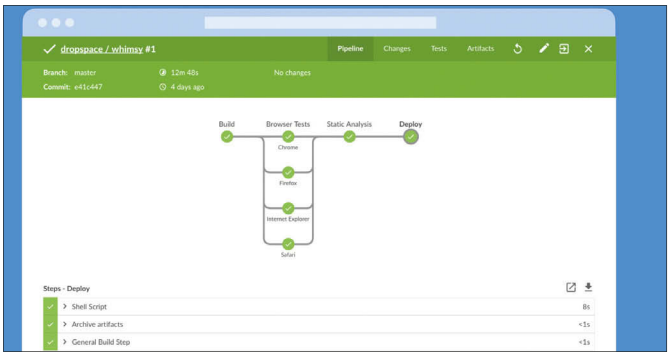
L'équipe de développement peut analyser ses propres opérations et affirmer qu'elle fait son possible pour améliorer ses processus, mais le reste du pipeline prendra toujours trop de temps si l'entreprise n'a pas vraiment analysé et amélioré son pipeline de livraison à un niveau plus global.

Imaginons qu'il faille trois jours entre l'application de modifications et la fin d'un test. Si le test échoue à la deuxième étape, vous pouvez configurer le build pour redémarrer à partir du jour deux, à la deuxième heure, soit juste avant l'erreur. Ne perdez pas de temps à refaire les mêmes tests. Sélectionnez juste le point de contrôle qui vous convient. Par exemple, si quelque chose ne fonctionne pas à cause d'un problème lors du test, le testeur peut choisir le point à partir duquel relancer le test.

Évitez de sur-automatiser

Plus un pipeline est compliqué à mettre en oeuvre, plus le temps et les coûts s'accumulent. La solution: automatiser le pipeline. Créez un workflow qui déplace automatiquement le build d'une étape à l'autre, en fonction de l'achèvement réussi d'un processus - ce qui permet de contrer l'effet des transferts délicats qui sont intégrés dans la conception du pipeline. Malheureusement, de nombreuses entreprises poussent la démarche à l'extrême. Dans une tentative d'automatisation des processus de développement, elles lient trop d'outils entre eux et finissent par sur-automatiser. Un tel niveau d'automatisation peut conduire à toutes sortes de résultats indésirables. Des problèmes peuvent survenir lors de contrôles qui ne peuvent pas être répétés par la personne effectuant la modification. Imaginons par





exemple qu'un processus soit programmé pour mettre à jour un site web dynamiquement et qu'un test échoue après quelques heures. Dans ce cas le testeur doit avoir les moyens de reproduire les étapes qui ont mené au message d'erreur pour analyser la situation par lui-même. Les pipelines doivent être configurés d'une manière qui permette l'intervention humaine. Tout l'intérêt d'un pipeline peut disparaître si l'échec d'un test vous oblige à complètement relancer un processus particulier. L'astuce consiste à trouver un équilibre entre sur-automatisation et sous-automatisation. Il s'agit de comprendre les exigences spécifiques à un projet en matière de développement, de test et de déploiement. Une fois que ces exigences sont clairement identifiées, il est temps de déterminer les processus et les outils d'automatisation les plus appropriés.

Conclusion

Les pipelines de déploiement continu sont comme des cultures végétales. Vous ne les semez pas pour les laisser ensuite à l'abandon : vous les cultivez pour qu'elles se renforcent et produisent de meilleures récoltes. Garder en tête ces quelques pratiques aidera vos équipes à créer de meilleurs logiciels plus rapidement, plus efficacement et de manière plus stratégique. Et par-dessus tout, elles apporteront une véritable valeur ajoutée à votre entreprise. Jenkins Pipeline vous permet de créer facilement ces types de pipelines sur Jenkins pour développer de meilleurs logiciels plus rapidement. •

Cloud : le problème de l'adhérence se pose toujours et encore

Une des constances de l'informatique est l'adhérence, la dépendance, le couplage fort. Quel que soit le terme, le résultat diffère finalement assez peu. Ce débat nous l'avons depuis nos débuts dans le monde informatique et il continue à hanter les méandres technologiques.

Dans un monde technique hétérogène où les couches se multiplient avec x fournisseurs pour le même service, il faut savoir rester calme. Un autre constat est éloquent. Il y a encore 3 ans, les plateformes Cloud étaient suffisamment limitées pour pouvoir être maîtrisées, aujourd'hui, une telle chose est impensable. Amazon, Google, Microsoft, Oracle, IBM proposent chacun des centaines de services en IaaS, PaaS, SaaS. Même maîtriser une seule plateforme relève de l'exploit.

Une des promesses du Cloud Computing était d'exporter de la flexibilité, des services à la demande faciles à provisionner et déprovisionner. Mais rapidement, la question de la réversibilité s'est posée et elle devient cruciale avec un usage massif de services cloud, à tous les niveaux.

Aujourd'hui, on pose la question du cloud natif, du multi-cloud, du cloud hybride. Des questions que l'on se posait déjà il y a 5 ans mais sans que cela ne pose de problèmes. Sommes-nous verrouillé avec le fournisseur cloud après l'avoir longtempts été sur le serveur et les postes de travail locaux ?

Quand on creuse un peu la question de la non-adhérence technique, tout le moins, d'une minimisation des dépendances, on s'aperçoit rapidement que ce n'est pas forcément aussi simple qu'il y parait : l'indépendance a un coût (et pas uniquement en €).

Il faut aussi regarder la latence, le SLA, le coût. Car tous ces éléments changent d'un fournisseur à un autre.

Mais le cloud permet de créer des environnements de développement / test très rapidement. On peut instancier un nouveau service très rapidement sans se soucier de la plomberie. Et ça ce sont déjà de gros avantages !

note rédaction

Le site Drupal propose un post intéressant sur la migration d'un site vers un autre fournisseur. La procédure n'est pas forcément compliquée mais elle peut prendre du temps : https://www.drupal.org/docs/7/backing-up-and-migrating-a-site/migrating-a-site

Prenons un cas courant : je migre un wordpress d'un fournisseur A à un fournisseur B

Exemple, si tu as un site Wordpress (WP) basé sur du MySQL et que tu souhaites changer de provider, il te faut faire un export de tes données (en créant un Dump) puis remonter ton site WP chez le nouveau provider en faisant bien attention à avoir la même version de PHP, de WP et de MySQL (idem avec Drupal). Selon mes différentes expériences, cela ne fonctionne malheureusement pas à tous les coups car la restauration du dump crée parfois des problèmes en écrasant le dump initié lors de l'installation de ton CMS. Une autre solution est de ne faire qu'un export des données du site via un plugin direct dans le site mais, et là encore, l'affaire n'est pas gagnée à 100% car le plugin peut être instable... Bref, il m'est arrivé une fois de devoir faire le transfert des data à la main pour un site Prestashop et il en résulte que je ne l'aime pas depuis car je pense que le problème était lié à l'export depuis le CMS.

Tout installer dans des containers semble une bonne alternative, mais tu ne restes pas libre de la techno sous-jacente. Un container Docker sous Linux doit (pour le moment) encore rester un conteneur sous Linux dans la nouvelle infra. Je n'ai pas encore testé le déplacement d'un container créé en mode CaaS par exemple vers une infra IaaS ou vice versa mais je m'attends à rencontrer des problèmes liés à la configuration réseau qui va relier tous mes containers. Seule une bonne orchestration de tout l'ensemble est incontournable pour cela, mais les spécialistes qui maîtrisent parfaitement le sujet restent rares car peu de recul sur ce type de migration en prod pour le moment.

Je conseille toutefois de dupliquer le site en question avant toute migration (attention : copie carbone parfaite, NDLR)

• Estelle Auberix

Philippe Charrière
(@k33g_org | philippe.charriere@clever-cloud.com) est Technical Evangelist et Responsable Commercial chez Clever Cloud (https://www.clever-cloud.com/)

Cloud Native?

C'EST JUSTE UNE HISTOIRE DE CONFIGURATION... OU PRESQUE.

Ces dernières années, dans le "buzzword cloud", dans le top 5, nous avons "cloud native" (ok, il y a aussi "serverless", "lambda", ...). L'objectif de cet article est un début (ou une tentative?) d'explication de ce qu'est "une application cloud native".

Je dis un "*début d'explication*" car la littérature sur le sujet commence à fleurir (Pivotal a écrit de très intéressantes choses sur le sujet, et notamment **Beyond the Twelve-Factor App** cf.http://www.oreilly.com/webops-perf/free/beyond-the-twelve-factor-app.csp). Par ailleurs, tout le monde a un avis sur le sujet... et pas forcément le même avis. Je vais donc tenter de faire simple et rester le plus pragmatique possible.

Une définition

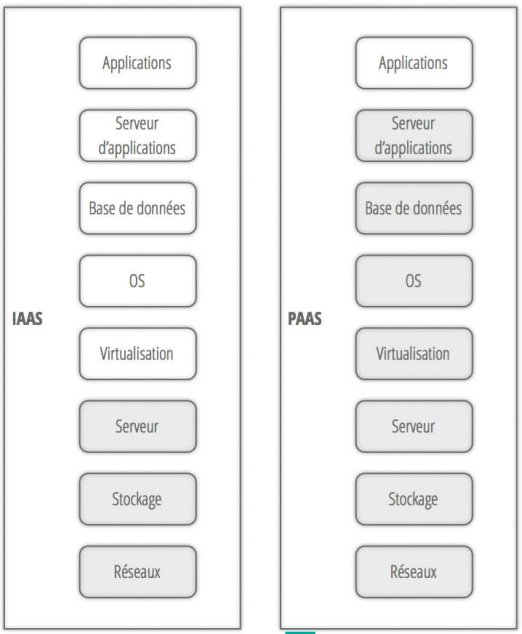
Commençons par une définition. Alors c'est la mienne, elle n'engage que moi, elle est simple, mais vous pouvez m'en parler ici https://github.com/k33g/q/issues ou sur Twitter (@k33g_org) si vous souhaitez la compléter, faire des remarques, dire que c'est n'importe quoi... Bref ...

Une application **cloud native** est une application :

- autonome (pas besoin de serveur d'application. Mais il y a des exceptions, toujours !) ;
- avec une configuration externalisée ;
- qui va fonctionner à l'identique dans tous les environnements (on oublie le "*mais, pourtant cela fonctionnait sur ma machine!*") et sans se préoccuper de ses dépendances ;
- et enfin, c'est une application que vous allez pouvoir déployer 100 fois dans la journée si vous en avez envie, et sans que personne ne s'en aperçoive ;
- c'est aussi une application, pour laquelle le développeur n'a à se soucier que de son code (et surtout pas de la partie infrastructure).

Mais comment fait-on une application "cloud native" ?

Sur le papier cela semble sympa, mais cela n'est-il pas compliqué de fabriquer une application **cloud native** ? En fait pas tant que ça, mais il y a quelques "petites" règles à suivre. Tout d'abord pour que cela fonctionne, il vous faut un **PaaS** (Platform as a service). Certains me diront qu'il est possible de le faire avec un **IaaS** (Infratructure as a service), soit, mais pourquoi s'embêter? De plus le cinquième item de ma définition ne serait plus valable. D'autres me rétorqueront, "*oui, mais nous, on fournit à nos équipes des VM préinstallées (ou du Docker) avec tout ce qui va bien dedans*", auxquels je répondrais : "*Ah oui? Tu as fait un PaaS à la mano donc?*".



1 *Responsabilités entre un IaaS et un PaaS*

Rapidement, le PaaS, qu'est-ce que c'est?

Donc, un PaaS (Platform as a Service) c'est une plateforme cloud pensée essentiellement pour les développeurs qui n'ont pas à se préoccuper de la partie infrastructure et système d'exploitation (en général lorsqu'un développeur s'attaque à cette partie, c'est le drame - pour rappel, je suis développeur aussi). Le PaaS va faire plein de choses pour vous : redonder, servir l'élasticité, du load balancing, sécuriser, faire des mesures, optimiser, ... (vous pouvez opter pour du IaaS, mais il va falloir s'en occuper). 1

Sur le schéma vous voyez en gris les services que vous apporte la plateforme. Vous voyez tout de suite qu'avec un PaaS, finalement vous n'avez à vous préoccuper que de votre application ; tout le reste est fourni. Alors, vous pouvez vous dire que dans le cas du PaaS, le fournisseur du PaaS possède/contrôle une grosse partie de la solution. Oui, c'est vrai, mais justement si votre application est **"cloud native"** et n'utilise pas de spécificités de la plateforme, vous avez la garantie de pouvoir la faire "tourner" où vous voulez.

Ma première application "cloud native"

Maintenant que le paysage est posé, voyons comment coder notre application cloud native (rien de plus simple). Un des précurseurs des PaaS orientés développeurs fut Heroku et ils posèrent les bonnes pratiques pour faire une bonne application pour le cloud avec ce que l'on appelle les **12 Factors** (https://12factor.net/). Nous n'allons pas passer les 12, mais "bosser" avec certains qui vont nous garantir de déployer et scaler facilement notre webapp (en plus, il peut y avoir d'autres critères que ces 12 facteurs). Je vous parlerai des 5 bonnes pratiques que je trouve essentielles (ensuite allez lire les autres facteurs, tout en sachant qu'il peut y avoir encore d'autres pratiques contribuant à la qualité de **"cloud native"**).

- 1 seule base de code ;
- Les dépendances ;

- La configuration ;
- Les services externs ;
- et enfin le port binding.

Selon moi, le meilleur type d'application représentative des applications **"cloud native"**, c'est le microservice. Pour mes exemples, j'ai choisi Vert.x qui permet de développer facilement des micro-services. Mais en quoi un micro-service est il **cloud native**? Eh bien, le microservice :

- est "simple", il n'a qu'un seul objectif (du coup, le code sera moins compliqué) ;
- peut avoir plusieurs versions: plusieurs versions d'un même micro-service peuvent cohabiter dans un même environnement au même moment ;
- doit être facilement scalable, sans pour autant augmenter les ressources des autres microservices constituant une application ;
- doit pouvoir être facilement utilisable par d'autres applications.

Et donc le microservice se prête bien aux pratiques que je vais vous présenter.

1 seule base de code

Le code source d'une application **"cloud native"** doit être contenu dans un seul repository, mais donnera lieu à plusieurs déploiements de la même application (en test, en production, localement sur le poste du développeur). Dans le cas des micro-services qui peuvent exister en plusieurs versions, qui sont indépendants et qui peuvent être codés dans différentes technologies, cela prend tout son sens, notamment pour le build. Cela va simplifier les process. C'est aussi ce qui va simplifier la vie de votre serveur de CI pour générer un livrable identique en développement, en test ou en production. Tout le monde n'est pas forcément de cet avis, je vous conseille cet intéressant article sur le sujet (même si je ne suis pas d'accord : https://blog.shippable.com/our-journey-to-micro-services-and-a-mono-repository

En exemple, vous trouverez ici une organisation GitHub avec des microservices que j'utilise pour des démonstrations (et donc un repository par micro-service) : https://github.com/the-big-bang-theory.

Il y a un autre ENORME avantage à avoir un repository par application. En effet, certains PaaS comme Heroku ou Clever-Cloud permettent de déployer une application à partir de son code source (donc de son repository) par un simple git push et ensuite le PaaS se chargera de builder votre application avant de la déployer (cela permet de partir à chaque fois avec un environnement d'exécution "propre" et avec la garantie que vous avez déployé une application dont la compilation a été un succès - c'est idéal pour faire de déploiement continu sans efforts). Si vous passez dans un mode "mono repository", vous ne pourrez pas utiliser cette "facilité" et vos processus de déploiement seront "légèrement" plus compliqués.

Remarque importante

Dépendances

"*Une application cloud native ne dépend jamais de l'existence implicite de packages au niveau du système. Elle déclare toutes ses dépendances, complètement et exactement, à travers un manifeste de déclaration de dépendances*". Autrement dit, en tant que développeur, pensez votre application "autonome", c'est à dire qu'elle va embarquer avec elle toutes les dépendances nécessaires à son exécution quel que soit l'environnement d'exécution. C'est par exemple ce que vous allez faire avec Maven qui va vous permettre de déclarer vos dépendances, et ensuite Maven s'assura de vérifier ces dépendances et de construire un jar unique (uberjar ou fat jar) à déployer.

Votre application va embarquer son propre serveur http (si vous faites du service web) et n'aura donc nullement besoin d'un serveur d'application pour s'exécuter. Dans le cas de Vert.x, le framework embarque Netty, ce qui nous permettra de produire un jar totalement autonome qui une fois démarré exposera un service web sans aucune aide d'un quelconque serveur d'application. Une application Java + Vert.x est un bon candidat pour développer une application **cloud native** :

- elle est "accompagnée" d'un fichier de build Maven ou Gradle qui décrit toutes les dépendances nécessaires pour "packager" l'application ;
- Vert.x est basé sur Netty ce qui lui permet de fournir son propre serveur http.

Ceci est une partie du fichier pom.xml qui me permet de construire le livrable de mon microservice **Stuart** https://github.com/the-big-bang-theory/stuart .

```
<project ...>
 <groupId>garden.bots</groupId>
 <artifactId>stuart</artifactId>
 <version>1.0-SNAPSHOT</version>

 <properties>
  <vertx.version>3.5.0</vertx.version>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  <main.verticle>garden.bots.Stuart</main.verticle>
 </properties>

 <build>
  <plugins>
   <!-- foo -->
  </plugins>
 </build>

 <dependencies>
  <dependency>
   <groupId>io.vertx</groupId>
   <artifactId>vertx-core</artifactId>
   <version>${vertx.version}</version>
  </dependency>

  <dependency>
   <groupId>io.vertx</groupId>
   <artifactId>vertx-web</artifactId>
   <version>${vertx.version}</version>
  </dependency>

  <dependency>
   <groupId>io.vertx</groupId>
   <artifactId>vertx-service-discovery</artifactId>
   <version>${vertx.version}</version>
  </dependency>

  <dependency>
   <groupId>io.vertx</groupId>
   <artifactId>vertx-service-discovery-backend-redis</artifactId>
   <version>${vertx.version}</version>
  </dependency>
```

```
</dependencies>
</project>
```

Le fichier complet est ici : https://github.com/the-big-bang-theory/stuart/blob/master/pom.xml.
C'est ce qui va me permettre (ainsi qu'à mon serveur de CI) avec une simple commande mvn clean package d'obtenir un seul jar, complètement autonome, que je pourrais exécuter localement ou ailleurs de cette façon : java -jar target/stuart-1.0-SNAPSHOT-fat.jar.
C'est donc ce mécanisme qui vous permettra à partir du repository de produire automatiquement votre jar et de rendre ce processus reproductible (avec le serveur d'intégration continue).
Une autre bonne pratique est d'externaliser sa configuration (tout ce qui peut changer entre 2 déploiements).

Configuration

La configuration de votre application doit être externalisée. C'est en fait tout ce qui est susceptible de changer d'un environnement d'exécution à un autre: URL de base de données, port http, credentials, ... Une bonne pratique pour cela est d'utiliser les variables d'environnement. Continuons avec le même microservice https://github.com/the-big-bang-theory/stuart.
Je dois fournir des informations qui vont "aider" mon microservice à s'identifier dans un annuaire de microservices, ou un outil de type API Gateway pour expliquer comment le contacter, le port http qu'il expose (on est dans une VM ou un container), son nom, ... Je vais donc écrire un code de ce type pour extraire ces information des variables d'environnement :

```
String serviceName = Optional.ofNullable(System.getenv("SERVICE_NAME")).orElse("John Doe");
String serviceHost = Optional.ofNullable(System.getenv("SERVICE_HOST")).orElse("localhost");
Integer servicePort = Integer.parseInt(Optional.ofNullable(System.getenv("SERVICE_PORT")).orElse("80"));
String serviceRoot = Optional.ofNullable(System.getenv("SERVICE_ROOT")).orElse("/api");
```

Donc maintenant, je vais pouvoir démarrer mon microservice de cette manière sur mon poste :

```
SERVICE_NAME=stuart \
SERVICE_ROOT=hey \
PORT=8080 \
SERVICE_PORT=8080 \
java -jar target/stuart-1.0-SNAPSHOT-fat.jar
```

Et je pourrais l'appeler de cette façon : http://localhost:8080/hey.
Ou de cette façon sur un Paas :

```
SERVICE_NAME=stuart \
SERVICE_ROOT=hey \
PORT=8080 \
SERVICE_PORT=80 \
SERVICE_HOST=stuart.cleverapps.io \
java -jar target/stuart-1.0-SNAPSHOT-fat.jar
```

Et je pourrais l'appeler de cette façon : http://stuart.cleverapps.io/hey.
Donc je peux bien, à partir d'un même livrable, exécuter mon microservice de différentes façons dans différents environnements. Le code complet est ici : https://github.com/the-big-bang-theory/stuart/blob/master/src/main/java/garden/bots/Parameters.java#L40.

Mais voyons comment cette pratique peut nous aider pour la pratique suivante.

Services externes (ou backing service)

Dans le contexte PaaS, un service externe correspond à la base de données, à un broker de message, à un système de cache, de storage, ... On parle aussi de ressources attachées qui peuvent être locales (dans le même PaaS) ou externes (fournies par un tiers) comme par exemple une base MongoDb hébergée chez MongoDb. Mais votre application ne devra jamais faire de distinction entre une ressource locale ou externe : par exemple, le déploiement de votre application doit pouvoir simplement remplacer votre base de données locale par une base de données hébergée chez un tiers. Le rattachement à une ressource doit pouvoir être changé facilement sans aucun changement de code. Là aussi, l'utilisation des variables d'environnement va faciliter cette pratique.
Dans mon cas, mon micro-service se connecte à un **"backend discovery"** qui est une base Redis (donc une ressource externe). Je vais donc avoir un code qui va ressembler à ceci :

```
ServiceDiscoveryOptions serviceDiscoveryOptions = new ServiceDiscoveryOptions();

// Redis settings with the standard Redis Backend
Integer redisPort = Integer.parseInt(Optional.ofNullable(System.getenv("REDIS_PORT")).orElse("6379"));
String redisHost = Optional.ofNullable(System.getenv("REDIS_HOST")).orElse("127.0.0.1");
String redisAuth = Optional.ofNullable(System.getenv("REDIS_PASSWORD")).orElse(null);
String redisRecordsKey = Optional.ofNullable(System.getenv("REDIS_RECORDS_KEY")).orElse("vert.x.ms");

return ServiceDiscovery.create(
 vertx,
 serviceDiscoveryOptions.setBackendConfiguration(
  new JsonObject()
   .put("host", redisHost)
   .put("port", redisPort)
   .put("auth", redisAuth)
   .put("key", redisRecordsKey)
 ));
```

Ce qui me permettra plus tard d'aller par exemple m'enregistrer dans le backend **"discovery"** (et aussi de l'interroger) :

```
discovery.publish(record, asyncResult -> {
 if(asyncResult.succeeded()) {
  System.out.println("The Microservice is published! " + record.getRegistration());
 } else {
  System.out.println("Not able to publish the microservice: " + asyncResult.cause().getMessage());
 }
});
```

Une fois encore, la pratique de la configuration externalisée va me permettre finalement d'utiliser n'importe quelle base Redis dans les environnements de mon choix. Comme ici sur mon poste pour tester :

```
SERVICE_NAME=stuart \
SERVICE_ROOT=hey \
PORT=8080 \
```

```
SERVICE_PORT=8080 \
REDIS_HOST=127.0.0.1 \
REDIS_PORT=6379 \
java -jar target/stuart-1.0-SNAPSHOT-fat.jar
```

Ou bien sur mon PaaS préféré :

```
SERVICE_NAME=stuart \
SERVICE_ROOT=hey \
PORT=8080 \
SERVICE_PORT=80 \
REDIS_HOST=b3bavzwrj-redis.services.clever-cloud.com \
REDIS_PORT=3033 \
REDIS_PASSWORD=pandas-are-great \
java -jar target/stuart-1.0-SNAPSHOT-fat.jar
```

Remarque

je pourrais même utiliser cette commande en local pour faire connecter mon microservice local à la base de Redis du PaaS (ou à toute autre base Redis).

Là, une fois encore, on note que la pratique de la configuration externalisée va grandement nous aider pour les déploiements, la mise en oeuvre de solutions de haute disponibilité, les tests, ... Le code complet est ici : https://github.com/the-big-bang-theory/stuart/blob/master/src/main/java/garden/bots/Parameters.java#L13.

Passons à la dernière pratique, qui est en partie une redite de ce que j'expliquais dans le paragraphe **Configuration**.

Association de port (ou port binding)

Une application **cloud native** est autonome et ne doit pas dépendre d'un serveur d'application (il peut y avoir des exceptions mais dans ce cas-là il devra y avoir une corrélation 1:1 entre l'application et le serveur d'application, autrement dit, on n'exécutera jamais plusieurs applications dans un même serveur d'application). En fait le bénéfice de ceci est assez simple à comprendre : votre application peut être aussi bien exécutée en local sur votre poste et accessible sur http://localhost:8080 que sur le serveur de tests http://192.168.1.10:9090 ou sur le serveur de production http://hello.world.com. Et une fois encore, tout cela sans changer une seule ligne de code, mais en utilisant à nouveau les variables d'environnement. C'est aussi ce qui va permettre à votre PaaS de gérer la scalabilité horizontale de votre application, c'est à dire augmenter le nombre d'instances, puis faire du load balancing dessus. Cela veut signifier aussi que chacune de vos applications devient aussi potentiellement une ressource externe utilisable par d'autres (pensez API first) et que vous pouvez facilement les instancier plusieurs fois sur une même plateforme pour, par exemple, proposer le service en plusieurs versions, ou n fois le même service mais "pointant" sur des bases de données différentes, ... Mais toujours sans avoir à modifier quoi que ce soit dans le code. Code qui ressemblera à ceci pour démarrer le micro-service et où vous notez une fois de plus que l'utilisation de variables d'environnement est indispensable pour nous permettre cette dernière pratique :

```
class Parameters {
 /* this is just a part of the code */
 static Integer getHttpPort() {
  return Integer.parseInt(Optional.ofNullable(System.getenv("PORT")).orElse("8080"));
 }
}

Integer httpPort = Parameters.getHttpPort();
HttpServer server = vertx.createHttpServer();
server
 .requestHandler(router::accept).listen(httpPort, result -> {
  if(result.succeeded()) {
   System.out.println("Listening on " + httpPort);
  } else {
   System.out.println("Houston, we have a problem: " + result.cause().getMessage());
  }
});
```

Et voilà, vous avez une application déployable facilement sur n'importe quel bon PaaS digne de ce nom.

Existe-t-il des fournisseurs cloud native?

Je dirais **non**. Il existe des fournisseurs de PaaS qui vont vous apporter différents services vous permettant de mettre en oeuvre des applications **cloud native**. A vous d'utiliser ces services à bon escient en utilisant les bonnes pratiques dont nous parlions plus haut. Finalement, c'est vous qui fournissez le côté **"Cloud Native"**, et le fournisseur de la plateforme **"clés en mains"** pour faire tourner tout ça. Mon conseil : évitez le plus possible d'utiliser des spécificités du fournisseur, sinon vous risquez d'avoir une application **cloud native** qui ne tournera que chez un unique fournisseur, et le jour où vous voudrez changer, cela vous demandera quelques efforts de développement. Si je devais lister des solutions PaaS (publics et privés) à base de VM ou de containers permettant de faire ça, je dirais, dans le désordre, et vérifiez les runtimes proposés (Java, NodeJS, Ruby, PHP, Haskell, Rust ...) pour être sûr de déployer les technologies que vous utilisez :

- CloudFoundry https://pivotal.io/platform (public + privé) ;
- Heroku https://www.heroku.com/ ;
- Clever Cloud https://www.clever-cloud.com/ (public + privé) *disclaimer: je travaille chez Clever Cloud* ;
- OpenShift https://www.openshift.com/ (public + privé) ;
- AWS Elastic Beanstalk https://aws.amazon.com/elasticbeanstalk/ ;
- Google Cloud Platform https://cloud.google.com/appengine/ ;
- Dokku le mini PaaS que vous hébergez vous même (sympa pour s'entraîner) http://dokku.viewdocs.io/dokku/ ;
- Flynn que vous pouvez aussi facilement essayer en local https://flynn.io/ ;
- Zeit https://zeit.co/ ;
- Hasura https://hasura.io/ ;

Il y en a d'autres bien sûr. C'est terminé pour cette fois-ci. Je vous laisse prendre connaissance des 7 facteurs restants (https://12factor.net/), mais n'hésitez pas à lire le "free ebook" **Beyond the Twelve-Factor App** qui va un peu plus loin et qui est très instructif. Et n'hésitez pas non plus à me poser des questions par ici https://github.com/k33g/q/issues ou par mail. Une dernière chose: pour ceux qui voudraient "jouer" avec **Stuart** (mon microservice):

```
git clone git@github.com:the-big-bang-theory/stuart.git
cd stuart
mvn clean package
SERVICE_NAME=stuart SERVICE_ROOT=hey PORT=8080 java -jar target/stuart-1.0-SNAPSHOT-fat.jar
```

https://github.com/the-big-bang-theory.

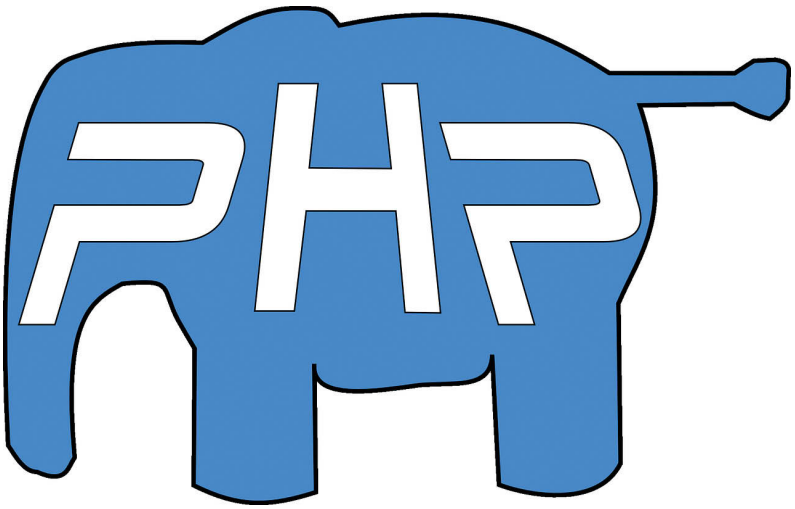
Remarque

N'oubliez pas de démarrer une base Redis avant. Vous trouverez d'autres exemples dans l'organisation

Gaëtan Cottrez
Chef de projets chez Orditech SA (Tournai, Belgique)
www.apprendre-la-programmation.net
Passionné d'informatique dès le plus jeune âge, c'est le développement, plus précisément le web, qui m'a toujours attiré. Analyser un besoin réel pour le développer soi-même dans le but de faire gagner du temps et d'automatiser est quelque chose de très excitant et passionnant pour moi.

Migrer son application vers PHP 7

C'est peut-être une des choses les plus redoutées par les développeurs, mais c'est tellement passionnant à faire et si instructif. Quand on a une application en production datant de plusieurs années, on a le devoir de la maintenir et si c'est nécessaire de la migrer que ce soit au niveau « hardware » autant qu'au niveau « software ».



La première application web que j'ai créée c'est l'ERP de la société dans laquelle je travaille. Elle est utilisée quotidiennement par l'ensemble des collaborateurs de la société soit une vingtaine d'utilisateurs. Malheureusement, celle-ci commence à souffrir de quelques ralentissements et de problèmes de performances. Il est donc nécessaire de remédier à cela.

Malgré le fait que je sois Chef de Projets depuis presque 3 ans, j'ai toujours la casquette de « web Application Developer » que je porte fièrement depuis plus de 7 ans en milieu professionnel. L'ERP en PHP/MySQL que je compte migrer a 7 ans depuis quelques mois et il n'a jamais réellement subi de refonte au niveau du code ni même d'upgrade au niveau technologie (nous sommes toujours en PHP 5.5).

Naturellement, des soucis commencent à se faire sentir : lenteurs dans certaines parties de l'application, base de données grandissante ou limites au niveau des technologies employées. Il y a un an, nous avons migré sur une version de MySQL plus récente (5.5 vers la 5.7) avec un serveur de bases de données dédié à l'ERP (la base de données était hébergée sur un serveur mutualisé qui contenait les bases de données de nos clients).

En plus de bénéficier des correctifs et de nouvelles fonctionnalités, nous avons observé de meilleures performances au niveau des exécutions et du traitement des requêtes. L'ERP a récupéré un petit coup de souffle.

Hélas, cela ne suffisait pas à optimiser complètement l'application comme on l'aurait souhaité. Heureusement, nous avions d'autres idées en réserve pour continuer notre épopée sur le chemin de l'optimisation :

- Migrer l'application vers PHP 7 ;
- Retravailler certaines requêtes SQL dans l'application présentant des latences ;
- Convertir petit à petit certaines fonctionnalités vers des technologies plus rapides et plus performantes.

Nous avons entrepris les 2 premières idées en parallèle, et l'autre a été gardée en réserve. Je vais rester concentré sur la partie migration vers PHP 7 puisque c'est celle-là qui nous intéresse.

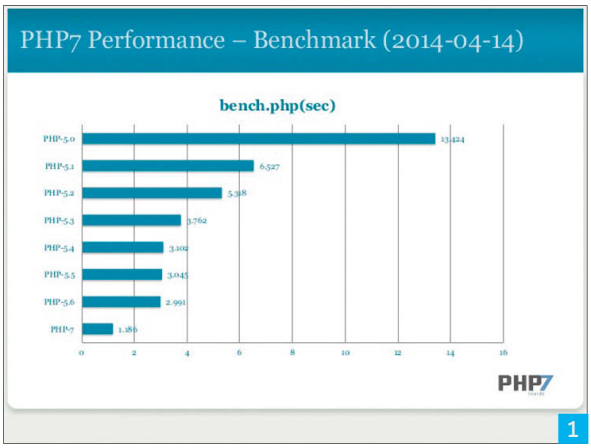
Pourquoi migrer vers PHP 7 ?

Il y a plusieurs raisons à citer, mais j'en prendrais 2 très légitimes.

La première raison c'est le côté nouveautés et performances du nouveau moteur.

Il faut bien comprendre que PHP est resté figé pendant de nombreuses années et le plus flagrant se manifeste de la version 5 pour arriver à la 7 :

- PHP 3 : 1998
- PHP 4 : 2000
- PHP 5 : 2004
- PHP 5.3 : 2009
- PHP 5.4 : 2012
- PHP 5.5 : 2013
- PHP 5.6 : 2014
- PHP 6 : début du développement en 2005 pour être finalement abandonné en 2010
- PHP 7 : 2016



1

L'échec de PHP 6 en est la principale cause et même s'il y eut des bonnes évolutions entre PHP 5.3 et PHP 5.6, il n'en restait pas moins que le langage traînait dernière lui un moteur vieillot et qui a commencé à ne plus être aussi performant qu'à l'époque. Ces mini-versions n'ont fait que maintenir le moteur et entretenir l'espoir d'un renouveau pour PHP.

C'était un pari risqué, mais réussi ! Les benchmarks pour PHP 7 que l'on peut trouver sur internet montrent des performances vraiment intéressantes sur ce nouveau moteur : 1

Mais on peut se poser la question : « est-ce vraiment le cas dans la réalité » ? Pour l'avoir testé par curiosité, je peux vous dire que vous allez voir une nette différence au niveau du chargement de vos pages pour

n'importe laquelle de vos applications PHP. Grâce à son nouveau moteur, PHP 7 est capable, de faire beaucoup plus de traitements en beaucoup moins de temps. La compilation de votre résultat pour être affiché à l'écran est devenue nettement plus rapide et surtout elle se fait en une seule fois.

C'est d'ailleurs ce dernier point qui est très important. Pour mon cas avec l'ERP en PHP 5.5 nous avons observé des latences de chargements sur certaines de mes pages sur l'ERP. Est-ce que la partie cliente est à mettre en cause (dans notre cas JQuery) ? Nous avons d'ailleurs pensé à migrer sur des technologies plus récentes et plus performantes comme « Angular » ou « React » pour tenter de pallier ce problème. Toutefois, lorsque l'on a réalisé un test rapide avec PHP7, nous avons tout de suite été convaincus que migrer vers ces technologies n'allait pas du tout arranger notre problème, car en réalité c'est bien la partie serveur qui en est la cause.

La deuxième raison c'est tout simplement la mise en place des bonnes pratiques et des meilleures normes en termes de programmation. PHP n'est pas un très bon élève. C'est un des rares langages de programmation où vous pouvez vous abstenir complètement des bonnes pratiques par exemple concernant le typage de vos variables, fonctions ou méthodes des classes. Même si PHP 7 n'oblige pas complètement encore à le faire (heureusement sinon on serait obligé de refactoriser la totalité de nos applications), le langage ouvre une porte aux programmeurs pour les utiliser petit à petit afin de mieux sécuriser son code et d'éviter des futurs problèmes. PHP pourra être conseillé dans un avenir très proche comme un langage de programmation vous donnant les bonnes habitudes et les bonnes pratiques en termes de programmation.

Comment migrer en PHP 7 ?

Je vais être honnête avec vous. Plus votre projet contiendra de lignes de code, plus votre migration sera longue. Comme vous allez le voir dans mon exemple, la migration a été très fastidieuse.

La nouvelle version de PHP 7 a donné naissance à de nouvelles fonctionnalités comme le typage au niveau de vos fonctions ou méthodes, un nouvel opérateur ou encore de nouvelles syntaxes d'écritures de test plus légères qu'auparavant. Je ne rentrerai pas dans les détails puisque les nouveautés ont déjà été présentées dans des précédents numéros de ce magazine et que celles-ci ne nous intéressent pas dans notre cas.

Mais comme je l'ai dit un peu plus haut, il y a eu des mini-versions dans la version 5 de PHP. Beaucoup de fonctions ont été déclarées « DEPRECATED » et peu d'entre elles ont été déclarées obsolètes au fil des versions. La sortie de PHP 7 en a donc profité pour faire le nettoyage des « DEPRECATED ».

Et si vous êtes comme moi, vous vous êtes peut-être dit que « tant que cela fonctionne je laisse comme ça », même si vous saviez que cela deviendrait obsolète un jour ou l'autre. Le problème c'est que lorsque vous allez migrer votre application vers PHP 7 vous aurez droit à une erreur fatale. Et je peux vous dire que la liste de fonctions obsolètes est tellement longue que les citer ici n'aurait aucun intérêt.

Il y a également certaines conventions de nommages qui ne sont plus supportées comme les anciens constructeurs PHP 4, et je suis sûr que vous devez avoir quelques vieilles librairies qui en utilisent encore si vous ne les avez pas mises à jour.

Cela fait pas mal de facteurs qui feront que votre code ne sera pas fonctionnel sur PHP 7. Alors, comment peut-on vérifier que notre code est compatible PHP 7 ? En utilisant des outils prévus pour cela. L'un des plus connus est « Phan », mais cela fait partie d'une de ses nombreuses fonctionnalités. J'ai préféré en prendre un qui est dédié à cela, facile à installer et à utiliser.

PHP 7 Compatibility Checker

J'ai choisi l'outil PHP7cc qui fait très bien son travail. Comme son nom l'indique, il va analyser les fichiers PHP de votre projet et vous notifier les problèmes en temps réel sous forme de « warning » (en jaune) ou de « error » (en rouge).

C'est un parseur de codes, et donc naturellement, si vous avez des erreurs de syntaxe, il vous le dira également.

Le projet est disponible directement sur Github : https://github.com/sstalle/php7cc. Pour l'installer, il est possible de le rajouter comme dépendances à son projet via Composer. Personnellement, je n'ai pas utilisé cette méthode puisque je voulais utiliser cet outil juste pour me permettre de faire la transition vers PHP 7.

J'ai préféré utiliser un conteneur Docker déjà tout fait, notamment celui-là : https://hub.docker.com/r/ypereirareis/php7cc/.

Une fois celui-ci téléchargé, il suffit d'utiliser cette commande pour lancer le conteneur en liant notre projet : [docker-php7cc.sh]

Code à récupérer sur www.programmez.com

Et voici un extrait de mon résultat php7cc par rapport à notre ERP :

[php7cc-result.sh]

Le résultat va vous donner plusieurs informations utiles concernant les mesures que vous devrez prendre pour rendre compatible votre code avec PHP 7. Vous allez obtenir à chaque fois :

- le chemin et le fichier présentant une anomalie,
- une prévisualisation de la ligne de code concernée,
- l'erreur renvoyée par PHP 7 si vous tentiez d'exécuter le code.

Dans votre Shell, vous remarquerez très vite que certaines informations sont dans des couleurs différentes. Il y a 2 couleurs utilisées :

- Les lignes en jaune sont des « Warning » c'est-à-dire que votre code s'exécutera sur le nouveau moteur, mais rien ne garantit que le résultat attendu sera le bon ;
- Les lignes en rouge sont des « errors » c'est-à-dire que votre code va présenter une erreur fatale sous PHP 7 et il faut donc obligatoirement corriger la ligne de code.

Passons à l'action !

PHP7cc m'a indiqué tous les problèmes liés à mon application, il est temps de corriger les erreurs une à une et manuellement. Même si certaines d'entre elles peuvent être corrigées en masse, je vous conseille de vous attarder tout de même sur chacune d'elle pour la corriger correctement. Suivant le nombre de lignes de codes, cela prendra un certain temps. Le plus simple c'est de se préoccuper des « errors » avant les « warnings », car il faut bien commencer par quelque chose et bien évidemment les « errors » sont prioritaires.

Mettre à jour ses librairies et dépendances

La première chose à faire avant de commencer à mettre les mains dans le cambouis, c'est de mettre à jour toutes vos librairies et dépendances pour bénéficier de leur compatibilité avec PHP 7. Si pour « X » raisons vous ne pouvez pas mettre à jour vos librairies, vous n'aurez guère le choix de les adapter vous-même pour les faire fonctionner.

Les erreurs faciles à corriger

Vous devez ensuite traiter les différentes erreurs dans les fichiers de votre programme. Quand vous avez une erreur du genre « Removed fonction 'Nom de la fonction' called », il suffit de taper le nom de la fonction en question sur la documentation de PHP (php.net) et de trouver sa remplaçante. Voici un exemple avec la fonction « split »

On obtient l'erreur suivante :

[php7cc-split.sh]

Et voici ce que dit la documentation PHP : 2

J'ai choisi de la remplacer par « explode », car au niveau du passage des paramètres, ce sont les mêmes et j'ai surtout l'habitude de les utiliser.

On remplace cette ligne de code dans le fichier en question :

[php7cc-split-code.php]

Par celle-ci :

[php7cc-explode-code.php]

Une fois le fichier sauvegardé, il suffit de relancer la commande docker pour PHP7cc pour constater que l'erreur ne sera plus affichée dans le rapport.

Un autre exemple avec cette erreur « PHP 4 constructors are now deprecated ».

Voici l'erreur :

```
// La syntaxe de constructeur qui nous pose problème
function Numbers_Words_Locale_bg()
{
}
// remplacé par cette syntaxe de constructeur
function __construct()
{
}
```

En PHP 4 (et dans d'autres langages de programmation), on crée le constructeur d'une classe en créant une méthode du même nom que la classe.

Mais PHP 7 ne gère plus cette syntaxe et pour ce faire, il suffit de renommer la méthode avec la méthode magique « __construct » ce qui donne le résultat suivant :

[php7cc-constructor-code.php]

Ce sont 2 exemples d'erreurs qui peuvent être facilement résolus et de façon très rapide avec la documentation de PHP.

Des problèmes un peu plus complexes

Il y a des problèmes plus gênants à corriger. Le plus embêtant c'est lorsque vous avez une classe qui utilise un mot réservé en PHP 7. Et j'ai eu une fois le cas avec cette erreur :

```
File: /app/class/error.class.php
> Line 67: Class/trait/interface "Error" was added in the
global namespace
  class error
  {
  }
```

Je possède une classe nommée « error » et ce mot-clé est dorénavant réservé en PHP 7 pour un nom de classe. La solution pour résoudre ce problème c'est de modifier le nom de votre classe par un autre nom. Ce qui implique également de le changer partout dans l'application où elle est appelée. Dans mon cas, elle est appelée un peu partout. Le plus simple et le plus rapide pour faire cette manipulation c'est d'utiliser un outil pour cela. Heureusement, lorsqu'on utilise un IDE (PHPStorm dans mon cas), on bénéficie d'une magnifique fonctionnalité appelée « refactoring » qui va, en plus de renommer le nom de la classe, vous lister l'ensemble des fichiers qui l'utilisent et vous proposer de modifier l'ensemble de votre application pour modifier son appel automatiquement.

Bien entendu, ce n'est pas une valeur 100% sûre puisque j'ai très vite remarqué qu'il a renommé certaines choses par erreur. Par exemple, dans les fichiers traitants des uploads où j'exploite la donnée « error » du tableau $_FILES, celle-ci a été renommée en « errorLog » (le nouveau nom de ma classe). Mon code ne fonctionnait donc plus après cette refactorisation, c'est donc à utiliser avec prudence.

[php7cc-error-part-2.sh]

Un autre cas qui a été fastidieux et qui m'a pris pas mal de temps ce sont toutes les fonctions permettant de gérer les expressions régulières. J'en utilisais encore beaucoup dans l'application et un peu partout. Honte sur moi me direz-vous ! Les fonctions « ereg_replace », « ereg », « eregi_replace » et « eregi » sont devenues obsolètes. Heureusement, elles peuvent être remplacées par une équivalence avec quelques modifications :

- « ereg_replace » avec « preg_replace » avec le modificateur *i*
- « ereg » avec « preg_match » avec le modificateur *i*
- « eregi_replace » avec « preg_replace » avec le modificateur *i*
- « eregi » avec « preg_match » avec le modificateur *i*

Si vous en avez beaucoup à modifier comme cela a été mon cas, alors vous aurez du boulot. « preg_match » et « preg_replace » ont tous les 2 besoin de délimiteurs. Ce qui n'était pas le cas des anciennes fonctions. De plus pour « eregi » par exemple il faudra rajouter après votre fin de délimiteur un « /i » pour que cela

split

(PHP 4, PHP 5)

split — Scinde une chaîne en un tableau, grâce à une expression rationnelle

Avertissement Cette fonction est devenue *OBSOLÈTE* en PHP 5.3.0, et a été *SUPPRIMÉE* en PHP 7.0.0.

Les alternatives à cette fonction incluent :

- preg_split()
- explode()
- str_split()

2

fonctionne comme auparavant. Suivant vos « regex », cela sera du cas par cas.
Voici 2 exemples, dont un avec « ereg » et « ereg_replace » :

```
// ereg_replace ne fonctionne plus sur PHP 7
$FAN=strtoupper(ereg_replace('[^[:alnum:]\\?]', '', $VAT));
// On la remplace par un preg_replace avec des d√©limiteurs.
// ici ce sera des slashes /
$FAN=strtoupper(preg_replace("/[^[:alnum:]\?]/","",$VAT));


// ereg ne fonctionne plus sur PHP 7
$OK= ereg('(^[0-9]{1,8})([0-9]{2}$|\\?{1,2}$)', $FAN, $REGS);
// On la remplace par un preg_match avec des d√©limiteurs.
// ici ce sera des slashes /
$OK=preg_match("/(^[0-9]{1,8})([0-9]{2}$|\?{1,2}$)/", $FAN,$REGS);

// Imaginons que la fonction √©tait eregi
// Alors on aurait seulement √† rajouter un i ce qui donnerait
$OK=preg_match("/(^[0-9]{1,8})([0-9]{2}$|\?{1,2}$)/i", $FAN,$REGS);
```

Comme vous l'avez vu, certaines erreurs seront difficiles et longues à corriger alors que d'autres seront très simples et rapides.

Et ensuite ?

Vous vous doutez bien que corriger le code pour qu'il soit compatible PHP 7 n'est qu'une partie de la migration, vous allez devoir tester que les changements qu'effectués sont toujours fonctionnels. Et vous allez également devoir tester l'ensemble de votre application pour vérifier qu'il n'y a pas de comportement indésirable qui pourrait être interprété différemment sur le nouveau moteur de PHP.

Une remise en question

On en vient à la partie « testing » de notre application pour vérifier si tout va bien dans chacune des fonctionnalités de celle-ci. Et dans mon cas c'est le drame, car j'ai dû tout faire manuellement.
J'ai commencé l'ERP il y a 7 ans et je sortais de l'école. Je n'avais jamais réalisé ni même entendu parler de tests unitaires. Quand j'ai découvert ce que c'était, ainsi que leurs intérêts, je me suis dit ceci : « ça a l'air très sympa , mais je ne vois pas trop l'intérêt d'en mettre dans mon application. En plus, c'est long et ennuyeux à écrire donc on verra ça plus tard. » Vous devinez bien que le « plus tard » se transforme rapidement en « jamais ». Et cette migration m'a fait prendre conscience de mon erreur monumentale.
Si vous avez mis en place des tests unitaires dans votre code, alors cela sera certainement très simple de vérifier que vos adaptations n'ont pas vérolé certaines parties de votre application, et que l'ensemble du code est toujours fonctionnel.
Si vous n'avez pas de tests unitaires, alors vous n'aurez pas le choix que de tester chacun de vos changements manuellement, l'un après l'autre comme j'ai dû le faire.
C'est long, pénible et contre-productif. Mais j'en retire quand même une bonne expérience puisque cette situation m'a fait comprendre l'intérêt et la force des tests unitaires. C'est donc loin d'être une perte de temps d'en mettre dans son application. Mais sachez que dans tous les cas, vous devrez tester tout de même l'ensemble de votre application sur votre nouvel environnement PHP 7 comme test final pour valider définitivement vos changements. Les tests unitaires vous permettront de gagner énormément de temps et d'accélérer le test final.

Conclusion

PHP 7 promet réellement ce qu'il indique sur le papier : un langage très performant en exécution et surtout qui a su faire le ménage dans sa structure en respectant mise à jour après mise à jour un peu plus les standards et les bonnes pratiques en programmation. Il n'y a qu'à voir la sortie récente de la version 7.2 pour se rendre compte que le langage continue d'aller dans ce sens.
Concernant une migration vers PHP 7, suivant les outils que vous allez utiliser et la complexité de votre application, cela sera peut-être très simple et rapide pour migrer alors que pour d'autres ce sera l'inverse.
Mais une chose est sûre : notre devoir en tant de développeurs, c'est de maintenir et de migrer nos applications sur les dernières versions de nos langages quand elles sont utilisées tous les jours pour le meilleur confort de nos utilisateurs et pour nous donner une satisfaction personnelle. •

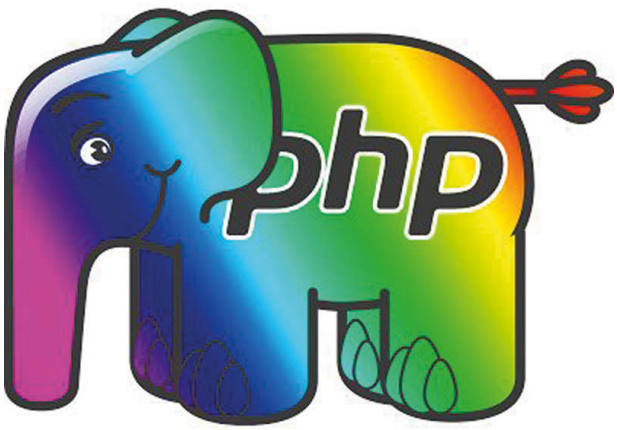
CHECK-LIST POUR LA MIGRATION

À faire :

- Créez une nouvelle branche sur votre repository git pour migrer vers PHP 7
- Utilisez un outil comme PHP7cc pour analyser chaque fichier PHP de votre projet afin d'identifier les lignes qui ne fonctionneront pas sur PHP 7
- Si vous trouvez des erreurs, résolvez-les !
- Relancez une nouvelle fois votre outil pour vous assurer que vous n'avez rien oubliez de corriger !
- Relancez tous vos tests unitaires, si vous en avez, pour vérifier que tout fonctionne correctement sur la nouvelle version PHP.
- Tester l'ensemble de votre application pour effectuer des tests réels et non des tests informatisés

À ne pas faire :

- Commencer à migrer votre projet sans vous êtes un minimum renseigné sur la nouvelle version
- Sous-estimez cette tâche en pensant que ce sera facile et rapide !
- Ne pas tester directement vos corrections au moment où vous les effectuez pour vous assurer que cela fonctionne
- Ne pas noter ce que vous faîtes et où vous en êtes ! Il est fort à parier que votre migration ne se fera as en une journée et encore moins plusieurs jours d'affilés en milieu professionnel
- Ne pas profiter de l'occasion pour remettre de l'ordre dans votre code lorsque vous effectuez une correction pour migrer vers PHP 7



Fabien Pigere
Après avoir travaillé pour de grands comptes, je suis devenu indépendant et spécialisé dans le front, tout particulièrement dans l'optimisation et l'algorithmie sur mobile et desktop.
fabienpigere@hotmail.com

Se débarrasser du back office avec Firebase !

Derrière ce titre un peu exagéré, je voudrais vous présenter Firebase. Créé à l'origine par Andrew Lee et James Templin, il fut racheté par Google en 2014. Il est disponible sur les plateformes Web, Android, iPhone,c++, Unity3d, Python, REST…

Cette API permet de gérer votre back-office sans trop d'efforts, nous allons par exemple voir comment authentifier l'utilisateur, stocker en base de données (format JSON de type arbre), stocker des documents (fichier, blob, JSON, base 64…), le tout avec un côté magique : le temps réel ! Que signifie ce temps réel ? Ce n'est pas argument commercial, mais une fonctionnalité "magique" dont on a du mal à se passer une fois qu'on y a goûté; nous verrons cela plus profondément sur la base de données. Nous allons voir comment l'utiliser dans sa version Web.

1 • S'inscrire

Il faut s'inscrire tout d'abord sur Firebase, et pour ça avoir un compte Gmail. Nous allons sur l'URL https://console.firebase.google.com/ et nous ajoutons un nouveau projet "ChatProgrammez".
Une fois dans le projet, nous allons autoriser les fonctionnalités suivantes pour la suite :
Allez dans authentification, puis Sign-in providers et autorisez Email, Phone et Google, cela nous servira plus tard. Vous devriez avoir le résultat suivant : 1
De même, il faut autoriser "database" et "storage".
Nous installons maintenant le CLI, c'est-à-dire l'outil de ligne de commande. Cette étape n'est pas obligatoire, mais nous évitera d'installer un serveur :

```
npm install -g firebase-tools
firebase login
firebase init
```

(PS : si vous n'avez pas npm, téléchargez node.js bien sûr)

À cette étape vous obtiendrez un répertoire "public" qui contient un index.html fonctionnel, il ne reste plus qu'a lancer le serveur :

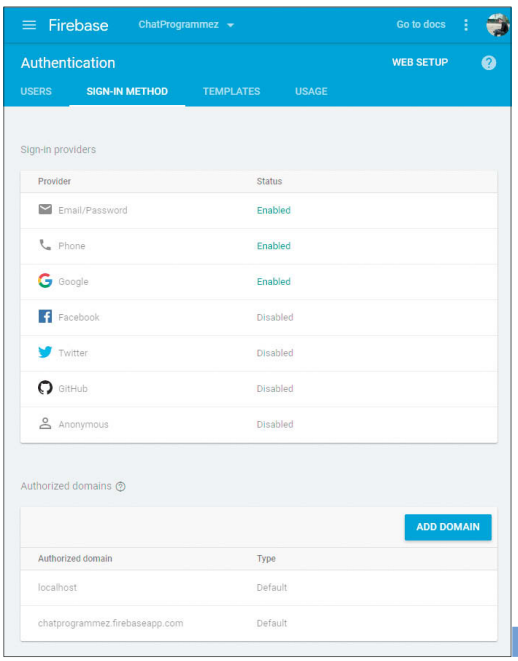
```
firebase serve
```

Qui vous indiquera le chemin nécessaire (localhost:5000 normalement).

2 • Authentification

Dans le back-office de tous les jours, l'authentification est une tâche obligatoire, mais pas très passionnante, il faut l'avouer ! De plus, de nos jours, les applications modernes offrent de multiples façons de se connecter (Facebook, Google+, Twitter), mais il faut aussi tenir compte des utilisateurs qui n'ont qu'un Email : il faut qu'ils puissent créer un compte, puissent changer de mot de passe, etc. Firebase nous aide à ce niveau en fournissant plusieurs méthodes très simples, qui permettent de gérer 99,99 % des besoins.
Il permet l'authentification par :

- Twitter
- Google+
- Facebook
- Email/mot de passe classique
- Validation du compte par SMS (et oui, rien que ça !)
- Github



1

Nous allons voir 3 types authentifications :
L'authentification par Email, la façon classique.
Dans le HTML on rajoute quelques contrôles.

```
<input id="email" type="email"/
<input id="password" type="password"/>
<button id="create">Create</button>
<button id="login">Login</button>
<button id="reset">Reset</button>
```

Le code JavaScript est sans surprise non plus :

```
$("#create").click(function ()
{
  var email = $("#email").val();
  var password = $("#password").val();
  firebase.auth().createUserWithEmailAndPassword(email, password).catch(function (error) {
    console.log(error);
  });
});

$("#login").click(function ()
{
  var email = $("#email").val();
  var password = $("#password").val();
  firebase.auth().signInWithEmailAndPassword(email, password).catch(function (error) {
```

```
        console.log(error);
    });
});



$("#reset").click(function ()
{
    var email = $("#email").val();
    firebase.auth().sendPasswordResetEmail(email).then(function () {
        // Email envoyé.
    }).catch(function (error) {
        console.log(error);
    });
});
```

À noter qu'il est possible d'envoyer un Email de vérification :

```
$("#create").click(function ()
    {
        var email = $("#email").val();
        var password = $("#password").val();
        firebase.auth().createUserWithEmailAndPassword(email, password).then(function () {
            var user = firebase.auth().currentUser;
            user.sendEmailVerification().then(function () {
                // Email sent.
            }).catch(function (error) {
                console.log(error);
            });
        }).catch(function (error) {
            console.log(error);
        });
    });
```

Authentification par Google+.

Pour commencer, allez dans la console, et autorisez l'authentification par Google. Puis avec la page HTML, ajoutez le code suivant :

```
    <button id="google">log</button>
    <img id="avatar"/>
```

et dans votre JavaScript :

```
    $("#google").click(function () {
        var provider = new firebase.auth.GoogleAuthProvider();
        firebase.auth().signInWithPopup(provider).then(function (result) {
            var token = result.credential.accessToken;
            var user = result.user;
            if (user != null) {
                photoUrl =;
                $("#avatar").attr("src", user.photoUR);
                var emailVerified = user.emailVerified;
            }
        }).catch(function (error) {
            console.log(error);
        });
    });
```

Voilà, vous avez fini l'authentification par Google+. Simple non ?

Authentification grâce au téléphone.

On rajoute quelques contrôles à notre page HTML :

```
    <button id="phone">log phone</button>
    <input id="tel" type="text" value="+3399999999"/>
    <input id="code" type="text" value=""/>
    <button id="btnCode">Code valide</button>
    <div id="captcha"></div>
```

L'authentification par téléphone nécessite un captcha, et firebase en fournit justement un (encore une chose de moins à faire !) :

```
    window.recaptchaVerifier = new firebase.auth.RecaptchaVerifier('captcha');
```

Puis le code simple suivant :

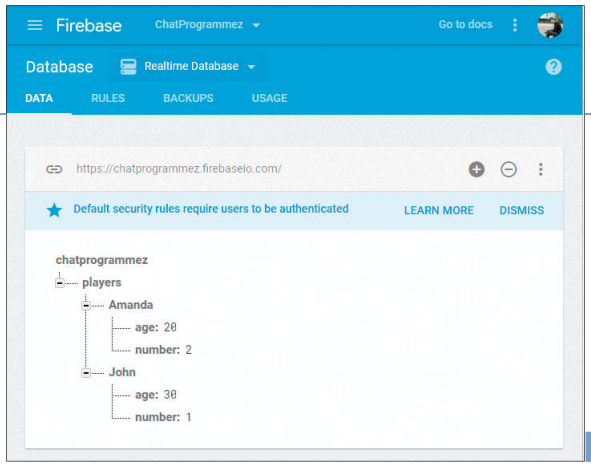
```
$("#phone").click(function () {
        var phoneNumber = $("#tel").val();
        firebase.auth().signInWithPhoneNumber(phoneNumber, window.recaptchaVerifier)
            .then(function (confirmationResult) {
                window.confirmationResult = confirmationResult;
                $("#btnCode").click(function () {
                    var code = $("#code").val();
                    confirmationResult.confirm(code).then(function (result) {
                        var user = result.user;
            // on est logué
                    }).catch(function (error) {
                        console.log(error);
                    });

                });
            }).catch(function (error) {
            console.log(error);
        });
    });
```

Bien sûr, il est possible de faire un logout à tout moment, avec le code suivant :

```
$("#logout").click(function ()
    {
        firebase.auth().signOut().then(function () {
            // succès
        }).catch(function (error) {
            console.log(error);
        });
    });
```

Et pour finir, une fonction simplifiant la gestion du SignIn :

```
firebase.auth().onAuthStateChanged(function (user)
    {
        if (user) {
            // User is signed in.
            var displayName = user.displayName;
            var email = user.email;
            var emailVerified = user.emailVerified;
            var photoURL = user.photoURL;
            var isAnonymous = user.isAnonymous;
            var uid = user.uid;
            var providerData = user.providerData;
            l("Sign in");
        } else {
            l("Sign out");
```



2

```
        // User is signed out.
        // ...
    }
});
```

Cette fonction est appelée à chaque changement d'utilisateur !
Je vous laisse regarder la documentation pour Facebook, Twitter et autres, c'est toujours aussi simple.

3 • La base de données NoSql temps réel.

Maintenant que nous sommes enfin logués, notre application veut stocker des données. La base de données de Firebase permet des requêtes simples sur des objets stockés en JSON.
Encore une fois, l'exemple sera simple et parlant de lui-même :

```
var playersRef = firebase.database().ref("players/");
        playersRef.set({
          John: {
            number: 1,
            age: 30
          },

          Amanda: {
            number: 2,
            age: 20
          }
        });
```

Le résultat est visible dans la console de la base de données : 2
Il existe un ensemble de fonctions permettant la lecture, écriture, update…
Là où cela devient intéressant, c'est qu'on peut installer des fonctions évènementielles sur la base de données à partir du client. C'est cela une base de données Realtime : vous pouvez être notifié sur un changement quelconque de la base de données ! Quelle est l'utilité ? Imaginons un logiciel de tchat : anciennement, le logiciel envoie une requête toutes les secondes pour vérifier si un message est disponible ou pas.
Dans notre cas, il suffit à Amanda de surveiller "players.amanda.messages" et elle sera informée seulement si un message arrive, quel que soit l'utilisateur qui l'a posté, ce qui est un gain évident de bande passante et de temps de réaction. Pour les curieux, en internet, Firebase utilise les WebSockets si disponibles. Voyons le code :

```
        var amandaRef = firebase.database().ref("players/Amanda/messages/");
        amandaRef.on('child_added', function (data) {
          console.log(data);
        });
```

À partir de là, construire une application de chat est assez évident.

Il nous faut deux champs :

```
    <input type="text" id="message"/>
    <div id="resultat">
```

Ainsi que deux fonctions :

```
    var messagesRef = firebase.database().ref("messages/");
    messagesRef.on('child_added', function (data) {
      var val = data.val().text;
      $("#resultat").append(val + "<br/>");
    });

    $("#message").change(function ()
    {
      var val = $("#message").val();
      var newmsg = messagesRef.push();
      newmsg.set({text: val});
    });
```

Et voilà, votre "IRC ligth" est opérationnel ! Besoin de rien d'autre; ouvrez deux browsers et tapez des messages, ils apparaissent automatiquement ! Magique non ?

4 • Le stockage d'objet lourd

Pour stocker les avatars, nous avons besoin d'autre chose qu'une base NoSql JSON, et là aussi, Firebase a une solution ; le Google Cloud Storage. Commençons par le HTML :

```
    <input id="file" type="file"/>
    <img id="img" />
```

Puis le code :

```
    $("#file").change(function ()
    {
      var file = this.files[0];
      var storageRef = firebase.storage().ref().child("file");
      storageRef.put(file).then(function (snapshot) {
        $("#img").attr("src", snapshot.metadata.downloadURLs[0]);
      });
    });
```

C'est simple n'est-ce pas ? On peut rajouter des méta-datas comme on veut , au format JSON (par exemple stocker un GUID d'un utilisateur?).

5 • Le déploiement.

Le déploiement est là aussi très simple, tapez en ligne de commande :

```
firebase deploy
```

Et voilà !

Conclusion

Nous n'avons vu que les fonctionnalités de base de Firebase. Je vous encourage vivement à étudier la documentation. Sachez qu'il existe des aides pour AdWord/AdMob afin de monétiser votre application ! Firebase permet aussi de créer des notifications sur votre mobile, d'utiliser Google Analytics dans votre programme Python ou iPhone, etc. Firebase simplifie énormément la création de sites Web, mais aussi de tout programme possédant un backend (en C++ par exemple), des applications portables, etc. Plus besoin de se préoccuper des tâches back-office de tous les jours, il le fait pour vous ! Il est à mon avis parfait pour une start-up ou une PME souhaitant un site très personnalisé. •

Damien VAN ROBAEYS
Ingénieur Systèmes Experteam
MVP Cloud & Datacenter Management
http://www.systanddeploy.com

GUI Metro Design avec WPF et PowerShell

L'arrivée de Windows 8/10 a entraîné de nombreux changements visuels et introduit ce que l'on appelle le « metro design ». Avec ce nouveau design les anciennes interfaces graphiques, que l'on pouvait créer avec Windows Forms, paraissent assez « old-school ».

Depuis le .Net Framework 3.0, en 2006, il est possible de créer facilement des GUI, en utilisant Windows Presentation Foundation (WPF).
La gestion de l'interface graphique, dans le WPF, se fait par le biais du langage XAML.
Un des avantages du WPF est la possibilité d'intégrer des Toolkits (ou thèmes) afin de donner à une interface basique [Fig. 1] un aspect plus moderne [Fig. 2], un aspect « Metro design ».

xaml, késako?

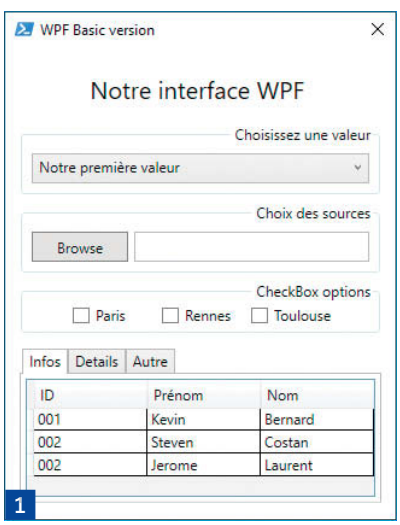
xaml est en langage de balises dérivé du *xml*.
Par exemple, l'affichage d'un bouton, suit la syntaxe suivante :

```
<button></button>
```

L'aspect du *control* est géré par l'ajout d'attributs. Ainsi pour donner un nom à notre bouton, il suffit d'ajouter l'attribut *Content*. Les attributs *Height* et *Width* jouent bien sûr sur les hauteur et largeur du bouton.

```
<Button Content="Browse" Height="20" Width="80"></Button>
```

Un des points forts du *XAML* est sa facilité de gestion par le *StackPanel*. Cette balise permet de regrouper plusieurs *Controls* « enfants », afin de gérer facilement leur emplacement dans une interface. Si l'on souhaite modifier la position de certains *Controls*, par exemple, afficher les *labels* l'un en dessous de l'autre dans l'interface [Fig. 3], il suffit de modifier l'attribut *Orientation* de *Horizontal* [Fig. 3] en *Vertical* [Fig. 4].



1

2

WPF et PowerShell

Le lien entre les fichiers xaml et ps1, s'établit par l'ajout de la partie suivante dans notre fichier PowerShell :

```
[void][System.Reflection.Assembly]::LoadWithPartialName('presentationframework')
function LoadXml ($global:filename)
{
  $XamlLoader=(New-Object System.Xml.XmlDocument)
  $XamlLoader.Load($filename)
  return $XamlLoader
}
$XamlMainWindow=LoadXml("Mon_Ficher.xaml")
$Reader=(New-Object System.Xml.XmlNodeReader $XamlMainWindow)
$Form=[Windows.Markup.XamlReader]::Load($Reader)
```

Controls et actions

Le lien entre un *Control* et son action dans le fichier PowerShell se fait en 3 étapes :

- Dans le fichier XAML : ajout de l'attribut **x :Name= "Mon_Bouton"**
- Dans le fichier PowerShell : déclaration du bouton comme ci-dessous :

```
$Mon_Bouton = $Form.FindName("Mon_Bouton").
```

L'action sur le bouton dans le PowerShell se fera via l'Event ***Add_Click***.

```
$Mon_Bouton.Add_Click({
  Votre code
})
```

Une GUI simple en WPF

L'interface graphique [Fig. 1] servira d'exemple pour la suite.
Ci-dessous le code *XAML* pour l'afficher :

```
<Window
  xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
  xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
  Title="WPF Basic version"
```

3

4

```
  Height="430" Width="340"
  ResizeMode="NoResize">
<Grid>
<StackPanel Orientation="Vertical" Margin="10,10,10,0" HorizontalAlignment="Center">
  <StackPanel Orientation="Horizontal"  FlowDirection="LeftToRight" Margin="0,5,0,0" Horizontal
Alignment="Center">
      <Label Content="Notre interface WPF " Margin="0,0,0,0" FontSize="20"/>
  </StackPanel>
  <GroupBox Header="Choisissez une valeur" FlowDirection="RightToLeft" Height="60"
 Margin="0,10,0,0" Width="300" HorizontalAlignment="left">
      <StackPanel Orientation="Horizontal"  FlowDirection="LeftToRight" Margin="0,5,0,0" Horizontal
Alignment="Center">
          <ComboBox x:Name="Choose_DS" SelectedIndex="0" Height="25" Width="280"
 Margin="0,0,0,0">
              <ComboBoxItem x:Name="value1" Margin="0,0,0,0">Notre première valeur
</ComboBoxItem>
              <ComboBoxItem x:Name="value2" Margin="0,0,0,0">Notre seconde valeur
</ComboBoxItem>
          </ComboBox>
      </StackPanel>
  </GroupBox>
  <GroupBox Header="Choix des sources" FlowDirection="RightToLeft" Height="60"
Margin="0,5,0,0" Width="300" HorizontalAlignment="left">
      <StackPanel Orientation="Horizontal" FlowDirection="LeftToRight" Margin="0,5,0,0"
 HorizontalAlignment="Center">
          <Button x:Name="Browse" Content="Browse" Height="25" Margin="0,0,0,0"
 Width="80"></Button>
          <TextBox x:Name="TextBox" Margin="5,0,0,0" Width="195" Height="25">
</TextBox>
      </StackPanel>
  </GroupBox>
  <GroupBox Header="CheckBox options" HorizontalAlignment="Left" Margin="0,5,0,0"
Height="45" FlowDirection="RightToLeft"  Width="300">
      <StackPanel Orientation="Horizontal"  FlowDirection="LeftToRight" Margin="0,5,0,0"
HorizontalAlignment="Center">
          <CheckBox x:Name="Paris" Content="Paris" Margin="4,0,0,0" Width="70"
Height="19"></CheckBox>
          <CheckBox x:Name="Rennes" Content="Rennes" Margin="4,0,0,0" Width="70"
Height="19"></CheckBox>
          <CheckBox x:Name="Toulouse" Content="Toulouse" Margin="4,0,0,0" Width="70"
Height="19"></CheckBox>
      </StackPanel>
  </GroupBox>
  <TabControl x:Name="Tab_Control" HorizontalAlignment="Left" Height="130"
VerticalAlignment="Top" Width="300" Margin="0,10,0,0">
      <TabItem Header="Infos" Margin="0,0,0,0">
          <Grid>
              <DataGrid SelectionMode="Single"  Name="DataGrid_XML" ColumnWidth="*"
ItemsSource="{Binding}"  Margin="2,2,2,2" >
                  <DataGrid.Columns>
                      <DataGridTextColumn Header="ID" Binding="{Binding ID}"/>
                      <DataGridTextColumn Header="Prénom" Binding="{Binding Prenom}"/>
                      <DataGridTextColumn Header="Nom" Binding="{Binding Nom}"/>
                  </DataGrid.Columns>
              </DataGrid>
          </Grid>
      </TabItem>
      <TabItem Header="Details" Margin="0,0,0,0">
      </TabItem>
      <TabItem Header="Autre" Margin="0,0,0,0">
      </TabItem>
  </TabControl>
</StackPanel>
</Grid>
</Window>
```

Plutôt sympa non ? Maintenant, comment rendre cette interface Metro design [Fig. 2] ?

Présentation de MahApps

MahApps (1) est un *toolkit* développé en C# qui permet de créer des interfaces WPF Metro Design (aspect Windows 10).
Créé par Paul Jenkins en 2011, MahApps est maintenu majoritairement par Jan Karger. Ce projet Open Source a connu un grand succès et a été téléchargé, à ce jour, 1million de fois sur GitHub (2). Windows 10 est un exemple parfait de Metro design. L'interface suivante [Fig. 5], est entièrement réalisée en *XAML* avec MahApps, disponible ici (3).

MahApps dans Visual Studio

Pour installer MahApps dans Visual Studio, lancez le gestionnaire de package Nuget puis tapez la commande ci-dessous :

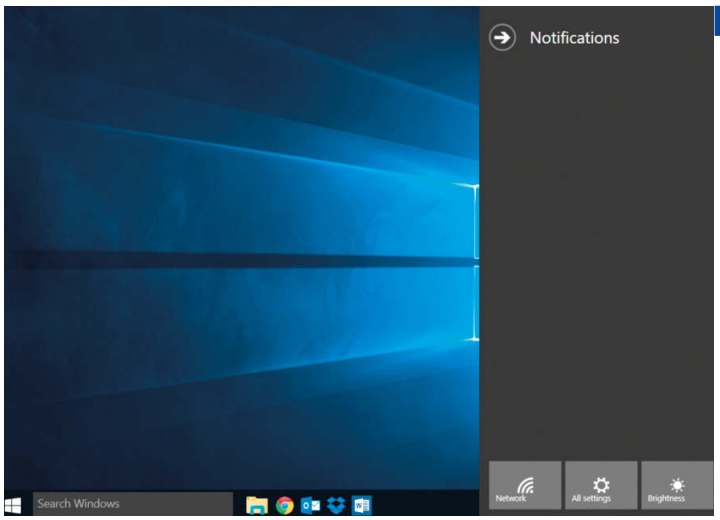
```
Install-Package MahApps.Metro
```

Afin de pouvoir utiliser MahApps dans PowerShell, il faudra récupérer des Assembly propres à MahApps.
Pour ce faire, il suffira de générer notre solution dans Visual Studio et récupérer la DLL : MahApps.Metro.dll

MahApps dans XAML & PowerShell

L'intégration du thème MahApps dans notre interface de test WPF se fera en deux étapes.

- Intégration de MahApps le fichier XAML, comme suit :
  - Remplacez les balises Window par metro:Window
  - Ajoutez les deux lignes ci-dessous dans la déclaration de notre Window, comme ci-dessous :
  - Ajoutez la partie ci-dessous, après la partie Window.



5 *Fake Windows 10 réalisé en WPF*

```
<Window.Resources>
  <ResourceDictionary>
    <ResourceDictionary.MergedDictionaries>
      <ResourceDictionary Source=".\resources\Icons.xaml" />
      <ResourceDictionary Source=".\resources\custom.xaml" />
      <ResourceDictionary Source="pack://application:,,,/MahApps.Metro;component/Styles/Controls.xaml" />
      <ResourceDictionary Source="pack://application:,,,/MahApps.Metro;component/Styles/Fonts.xaml" />
      <ResourceDictionary Source="pack://application:,,,/MahApps.Metro;component/Styles/Colors.xaml" />
      <ResourceDictionary Source="pack://application:,,,/MahApps.Metro;component/Styles/Accents/Cobalt.xaml"/>
      <ResourceDictionary Source="pack://application:,,,/MahApps.Metro;component/Styles/Accents/BaseLight.xaml"/>
    </ResourceDictionary.MergedDictionaries>
  </ResourceDictionary>
</Window.Resources>
```

Cette partie permettra, par exemple, de modifier le thème et la couleur principale des Controls de l'interface.
Dans le premier exemple, la couleur générale de notre interface est Cobalt et son thème BaseLight.

- Intégration de MahApps dans le fichier PowerShell. Pour cela il faut charger la DLL récupérée avant, en utilisant la ligne ci-dessous :

```
[System.Reflection.Assembly]::LoadFrom('assembly\MahApps.Metro.dll')| out-null
```

MahApps dans notre GUI

Les éléments étant un peu plus volumineux avec *mahapps*, une adaptation des tailles des **GroupBox**, est souvent nécessaire. Ci-dessous le code XAML correspondant à l'interface après ajout de MahApps [Fig. 2].
Code complet sur www.programmez.com

Une nouvelle couleur en un clic

Un des grands avantages de MahApps est sa flexibilité. Celle-ci nous autorise, entre autres, la modification globale des couleurs en modifiant l'unique ligne ci-dessous, sans le fichier XAML :

```
ResourceDictionary Source="pack://application:,,,/MahApps.Metro;component/Styles/Accents/Cobalt.xaml"/>
```

Par exemple, pour passer du bleu au rouge, il suffit de remplacer la valeur *Cobalt* par *Red* [Fig. 6].

```
ResourceDictionary Source="pack://application:,,,/MahApps.Metro;component/Styles/Accents/Red.xaml"/>
```

Changement de thème

Il est également possible de modifier le thème global de la fenêtre en lui donnant un aspect plus sombre.
Deux thèmes sont disponibles : *BaseLight* et *BaseDark*.
La modification du thème se fait via la ligne suivante :

```
<ResourceDictionary Source="pack://application:,,,/MahApps.Metro;component/Styles/Accents/BaseLight.xaml"/>
```

Dans notre 1er exemple, le thème utilisé est *Baselight*.
L'application du thème **BaseDark** se fait donc en remplacant **BaseLight** par **BaseDark** [Fig. 7].

```
<ResourceDictionary Source="pack://application:,,,/MahApps.Metro;component/Styles/Accents/BaseDark.xaml"/>
```

Si nous voulons appliquer le thème *BaseDark* avec des couleurs rouges, [Fig. 8], utilisons les lignes ci-dessous :

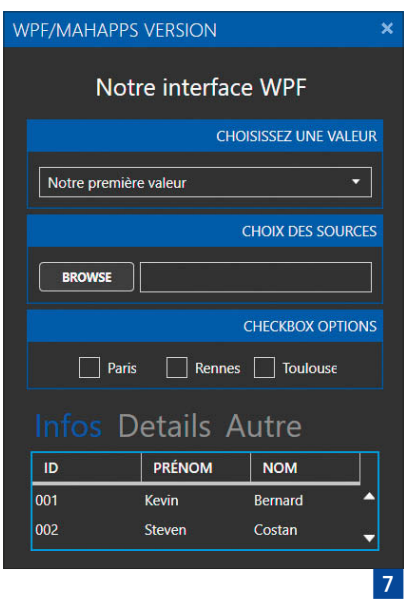
```
<ResourceDictionary Source="pack://application:,,,/MahApps.Metro;component/Styles/Accents/Red.xaml"/>
<ResourceDictionary Source="pack://application:,,,/MahApps.Metro;component/Styles/Accents/BaseDark.xaml"/>
```

Maintenant que nous avons vu l'aspect principal de notre interface suite à l'intégration de *MahApps*, nous allons voir quelques *Controls*, facilement exploitables.

Control Badge

Les badges sont présents nativement dans Windows 10 et permettent d'afficher un nombre d'éléments. Par exemple, dans Windows 10, ce badge [Fig. 9] permet de signifier 3 nouvelles notifications.
Ce control dans MahApps s'ajoute à un bouton [Fig. 10].
Ci-dessous, la partie à ajouter dans le fichier XAML, pour tel bouton :

6



7

8

```
<Controls:Badged x:Name="System_error" Width="150" Height="30" Badge="{Binding Path=BadgeValue}" Margin="0,15,0,0" BadgePlacementMode="TopRight">
  <Button x:Name="Button" Margin="0,0,0,0" Content="System event error" />
</Controls:Badged>
```

L'emplacement du badge sur le bouton peut facilement se modifier en utilisant l'attribut *BadgePlacementMode*.
Les valeurs disponibles pour cet attribut sont : *TopRight* [Fig.10], *TopLeft* [Fig.11], *BottomRight* [Fig.12], *BottomLeft*, *Top*, *Bottom*.
Dans PowerShell, la gestion du badge se fait par le biais de deux attributs. La première étape est bien sûr de déclarer le bouton et le badge.

```
$MyBadge = $Form.FindName("MyBadge")
$MyButton = $Form.FindName("MyButton")
```

La couleur du badge est gérée via l'attribut **BadgeBackground**, comme ci-dessous :

```
$MyBadge.BadgeBackground = "Blue"
```

La valeur du badge est gérée via l'attribut **Badge**, comme ci-dessous :

```
$MyBadge.Badge = "0"
```

L'exemple ci-dessous permet d'afficher, dans un badge associé à un bouton, le nombre d'erreurs systèmes rencontrées au cours des dernières 24h. Si aucune erreur n'a été rencontrée, le badge s'affiche en vert. Dans le cas contraire il s'affiche en rouge [Fig.13].

```
$Date = (Get-Date).AddDays(-1)
$System_Log = (Get-EventLog -LogName System -After $Date -EntryType Error).count
$MyBadge.Badge = $System_Log
If ($System_Log -ne "0")
  {
    $MyBadge.BadgeBackground = "Red"
  }
Else
  {
    $MyBadge.BadgeBackground = "Green"
  }
```

Control Tile

Un des control les plus parlants dans MahApps, quand on parle de Metro design, est le control Tile, ou tuile, boutons introduits depuis Windows 8.

Tile simple

L'exemple ci-dessous permet d'afficher deux **Tiles** simples [Fig.14].

```
<Controls:Tile x:Name="MyTile1" Background="Red" Width="125" Margin="3" Height="125"/>
<Controls:Tile x:Name="MyTile2" Background="#A200FF" Width="125" Margin="3" Height="125"/>
```

Pour modifier la couleur ou ajouter un titre il suffit de modifier, respectivement, les attributs **BackGround** et **Title** [Fig.15].

```
<Controls:Tile x:Name="MyTile1" Title="Messagerie" Background="#00a300" Width="125" Margin="3" Height="125"/>
<Controls:Tile x:Name="MyTile2" Title="Youtube" Background="#1BA1E2" Width="125" Margin="3" Height="125"/>
```

Tile avec icône

La tuile est un peu vide, ajoutons donc maintenant une icône [Fig.16], en utilisant le code ci-dessous :

```
<Rectangle Fill="White" Height="60" Width="60">
  <Rectangle.OpacityMask>
    <VisualBrush Stretch="Fill" Visual="{StaticResource appbar_mon_icone}" />
  </Rectangle.OpacityMask>
</Rectangle>
```

Un moyen simple de modifier l'icône est : 1) de se rendre sur le site : http://modernuiicons.com/ ; 2) d'y choisir une icône ; 3) de passer le curseur de la souris dessus [Fig.17].
Il suffira de remplacer *mon_icone* par la valeur affichée en remplaçant les « . » par des « _ ».
Le code XAML pour afficher nos tuiles [Fig.16] avec les icônes est le suivant :

```
<Controls:Tile x:Name="MyTile1" Title="Messagerie" Background="#00a300" Width="125"
 Margin="3" Height="125">
  <Rectangle Fill="White" Height="60" Width="60">
    <Rectangle.OpacityMask>
      <VisualBrush Stretch="Fill" Visual="{StaticResource appbar_office_outlook}" />
    </Rectangle.OpacityMask>
  </Rectangle>
</Controls:Tile>
<Controls:Tile x:Name="MyTile2" Title="Youtube" Background="#1BA1E2" Width="125"
Margin="3" Height="125">
<Rectangle Fill="White" Height="60" Width="60">
  <Rectangle.OpacityMask>
    <VisualBrush Stretch="Fill" Visual="{StaticResource appbar_youtube}" />
  </Rectangle.OpacityMask>
</Rectangle>
</Controls:Tile>
```

Tile avec image

Pour ajouter une image de fond à notre tuile, utilisez le code ci-dessous [Fig.18]

```
<Controls:Tile x:Name="MyTile1" Title="Desktop" Margin="3" Height="125" Width="147">
  <Controls:Tile.Background>
    <ImageBrush ImageSource="Images/desktop.jpg" />
  </Controls:Tile.Background>
</Controls:Tile>
```

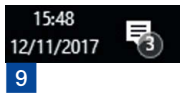
Affichage de Dialog

Le *dialog* permet, avec un aspect sympa, de remplacer les *msgbox*. Ne s'agissant pas d'un élément graphique affiché dans notre interface, ceux-ci sont ajoutés directement dans le fichier PowerShell, en tant qu'action d'un bouton.

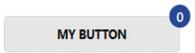
Dialog simple

Le code ci-dessous permet par exemple, d'afficher Dialog simple après avoir cliqué sur le bouton *Browse* [Fig.19].

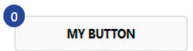
```
$Browse.Add_Click({
  [MahApps.Metro.Controls.Dialogs.DialogManager]::ShowMessageAsync($Form, "Hey :-)"
, "Plutôt cool non ?")
})
```



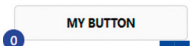
9



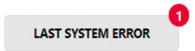
10



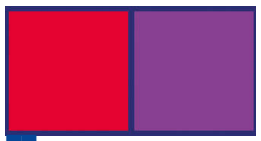
11

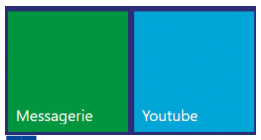


12



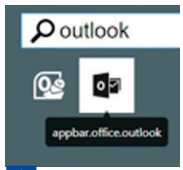
13


14

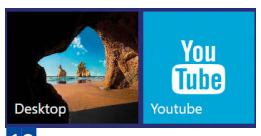


15

16



17



18

Dialog de confirmation

Nous pouvons également afficher un Dialog de confirmation [Fig. 20] avec le code ci-dessous :

```
$Browse.Add_Click({
  $ToBeOkOrNotOk = [MahApps.Metro.Controls.Dialogs.MessageDialogStyle]::Affirmative
AndNegative
  $Mon_Choix = [MahApps.Metro.Controls.Dialogs.DialogManager]::
      ShowModalMessageExternal($Form,"Hey","On continue ?",$ToBeOkOrNotOk)
  If ($Mon_Choix -eq "Affirmative")
  {#Code si bouton OK
  }
  Else
  {#Code si bouton Cancel
  }
})
```

Dialog de Login

Pour afficher un dialog permettant de saisir des identifiants [Fig. 21], utilisez le code ci-dessous :

```
$Browse.Add_Click({
  $MyCredentials = [MahApps.Metro.Controls.Dialogs.DialogManager]::ShowModal
LoginExternal($Form,"Login:","Type your credentials :)")
  $User_Login = $Login.Username
  $User_PWD  = $Login.Password
})
```

Plus d'informations sur l'utilisation de ce Control (4) avec PowerShell. Pour l'utilisation avec C# (5).

Le Control Flyout

Ce control présent dans Windows 8/10 à l'aspect suivant [Fig. 22].
L'ajout d'un Flyout se fait via le code ci-dessous :

```
<Controls:MetroWindow.Flyouts>
  <Controls:FlyoutsControl>
  <Controls:Flyout x:Name="Mon_Flyout" Header="Options" Position="Right" Width="200">
  <!-- Vos controls ici -->
  </Controls:Flyout>
  </Controls:FlyoutsControl>
</Controls:MetroWindow.Flyouts>
```

Le Flyout étant associé à un bouton, il faudra donc ajouter, dans PowerShell, l'attribut IsOpen, avec la valeur $True, au bouton permettant d'ouvrir le Flyout.

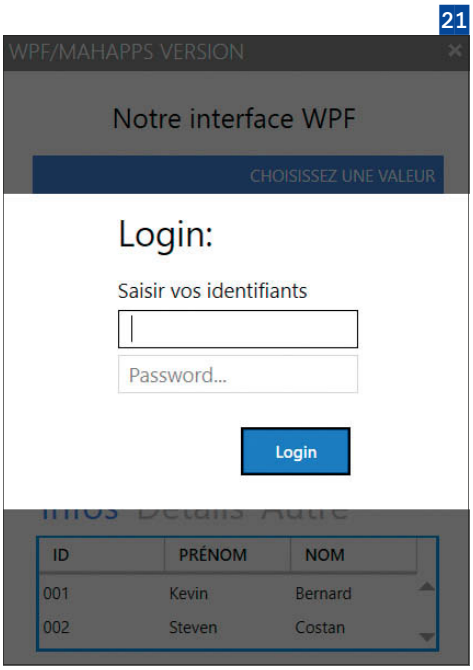
```
$Open_Flyout.Add_Click({
  $Mon_Flyout.IsOpen = $true
})
```

Le Control **Flyout** peut être affiché dans toutes les positions via la modification de l'attribut **Position**.
Les différentes valeurs possibles sont : **Right**, **Left**, **Bottom** [Fig. 22], **Top**.

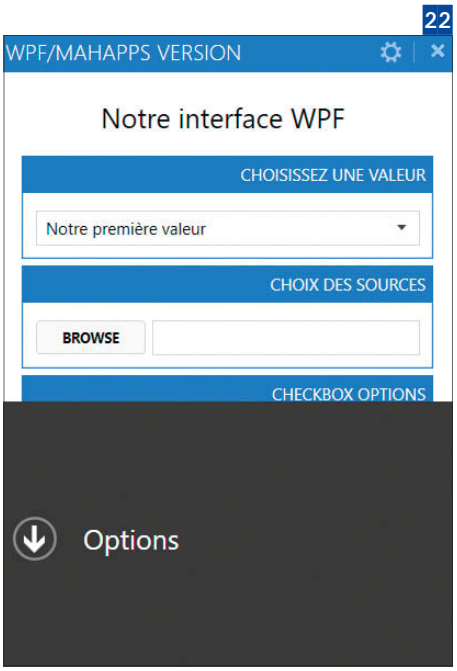
(1) http://mahapps.com
(2) https://github.com/MahApps/MahApps.Metro
(3) https://github.com/damienvanrobaeys/Windows10_Design_WPF
(4) https://dev4sys.blogspot.com/2017/07/using-mahapps-built-in-dialogs-with.html
(5) http://mahapps.com/controls/dialogs.html

19

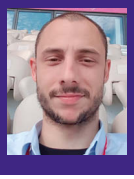
20



21



22



Sylvain Saurel
sylvain.saurel@gmail.com
Développeur Java / Android
https://www.ssaurel.com

Introduction au support HTTP/2 apporté par Java 9

Après 2 ans d'attente, Java 9 a finalement été lancé fin septembre 2017. Cette nouvelle mouture a fait les gros titres pour le système de gestion des modules qu'elle introduit. Il faut dire que la modularisation du JDK était une sorte de serpent de mer vieux de près de 10 ans. Néanmoins, Java 9 apporte également d'autres nouveautés présentant un intérêt majeur. Parmi celles-ci, nous allons nous intéresser plus particulièrement au support HTTP/2 accessible au travers d'une toute nouvelle API.

Le protocole HTTP/2 a été approuvé par l'Internet Engineering Task Force (IETF) en 2015, 16 ans après la sortie du protocole HTTP/1.1. Cette nouvelle version du protocole vient avec la promesse d'un temps de latence réduit tout en rendant également obsolète un certain nombre de solutions de contournements rendues nécessaires par HTTP/1.1 afin de répondre aux problématiques de temps de réponse nouvelles de notre époque.

Lancé en 1999, le protocole HTTP/1.1 a fait son temps et tout le monde est bien conscient que le Web a énormément changé depuis cette époque où il n'en était qu'à ses balbutiements. Les utilisateurs sont désormais de plus en plus impatients lorsqu'ils utilisent Internet, et si le temps de réponse dépasse ne serait-ce qu'une seconde, ils commencent déjà à se demander s'il n'y a pas un problème. Selon certaines recherches, le temps d'attention moyen sur une page au moment d'un clic ne dépasse pas 7 à 8 secondes. Ainsi, si une page met plus de 10 secondes à se charger, l'utilisateur aura depuis bien longtemps quitté le site considérant celui-ci comme hors d'usage.

Problématiques autour de HTTP/1.1

Cela signifie clairement que l'optimisation du temps de latence est essentielle. Parmi les différentes solutions de contournement nécessaires pour pallier les manques du protocole HTTP/1.1, on peut proposer la liste non exhaustive suivante :

- Une connexion HTTP ne pouvant télécharger qu'une ressource à la fois, les navigateurs effectuent des requêtes concurrentes dans le but de rendre les pages plus rapidement. Cependant, le nombre de connexions par domaine est limité. Pour pallier cette problématique, la technique du "domain sharding" a dû être mise en place. Concrètement, cela consiste à séparer les contenus sur différents sous-domaines.
- Une technique d'optimisation similaire a été imaginée pour combiner les différentes ressources utilisées par une page (CSS ou JavaScript) au sein d'un même bundle. Cela n'étant pas sans poser des soucis au moment où le navigateur doit sélectionner les ressources statiques pertinentes et les assembler.
- Le recours à la technique des Sprites d'Images pour packager au sein de mêmes bundles les fichiers CSS et JavaScript dans le but de diminuer le nombre de requêtes à réaliser entre le client et le serveur.
- Enfin, citons la technique de l'inlining des ressources CSS directement au sein du HTML ayant pour finalité de réduire le nombre de requêtes à effectuer.

Principaux avantages de HTTP/2

Fort du constat des différentes lacunes de HTTP/1.1, le protocole HTTP/2 a été conçu avec pour objectif premier de simplifier la gestion des infrastructures complexes. Alors que cette nouvelle version majeure du protocole HTTP est rétro compatible avec ce qui existe en HTTP/1.1, le protocole n'est plus basé sur du texte. Désormais, les échanges sont réalisés en mode binaire avec HTTP/2.

Le multiplexing qu'autorise HTTP/2 rend également, de facto, obsolètes les différentes solutions de contournement de HTTP/1.1 présentées précédemment. En effet, une connexion simple peut désormais gérer des flux multiples de manière bidirectionnelle. Ceci autorisant les clients à télécharger des ressources multiples de manière simultanée au sein d'une même connexion.

Basé sur le mode texte, HTTP/1.1 se révélait verbeux. Cela s'avérait d'autant plus problématique que les headers HTTP pouvaient être échangés à de multiples reprises. HTTP/2 va diminuer de manière drastique la consommation de bande passante en maintenant une table des headers HTTP au travers des différentes requêtes. Il s'agit essentiellement d'un mécanisme de déduplication et non d'un système de compression à proprement parler cependant.

Enfin, il paraît important de s'arrêter quelques instants sur la fonctionnalité push du protocole HTTP/2. De prime abord, certains pourraient penser qu'il s'agit d'un prolongement, voire d'une amélioration de ce que permettent les WebSockets. Néanmoins, il n'en est rien. Alors que les WebSockets permettent l'établissement d'une connexion full-duplex entre le client et le serveur dans le but d'autoriser le serveur à envoyer des données aux clients une fois la connexion TCP établie, HTTP/2 s'attelle à résoudre un problème différent. Le mode push de HTTP/2 concerne l'envoi des ressources aux clients de manière proactive sans que celles-ci ne soient demandées par le client. En pratique, cela signifie que le serveur, sachant qu'un site a besoin d'images, va toutes les envoyer d'un coup en avance de phase, bien avant que le client les requête. Leur affichage, le moment venu, sera donc instantané du point de vue du client.

Support de HTTP/2 dans Java 9

Compte tenu des nombreux avantages apportés par le protocole HTTP/2, un certain nombre de bibliothèques de codes Java proposaient déjà un support depuis un certain temps. Parmi celles-ci, on pourra citer : Jetty, Netty, OkHttp, Vert.x ou encore Firefly. Dans cet article, nous nous focalisons cependant sur le support de HTTP/2 fourni en standard par Java 9. Nous laisserons donc ces bibliothèques de côté. Le support de HTTP/2 est porté par le JEP 110 en charge de définir les exigences et les différentes évolutions de l'état du projet et de l'API dédiée. Intégré à Java 9 en standard, le projet demeure toujours en état d'incubation au sens du JDK, ce qui signifie en pratique qu'il ne remplace pas l'API UrlConnection utilisée jusqu'alors.

L'API dédiée au support de HTTP/2 sous Java 9 est placée au sein du package *jdk.incubator* . Pour utiliser l'API et tester ses différentes possibilités, il vous faudra donc dans un premier temps inclure le module *jdk.incubator.httpclient* au sein de votre projet :

```
module com.ssaurel.http2test {
  requires jdk.incubator.httpclient;
}
```

Création d'un objet HttpClient

L'API dédiée au support de HTTP/2 a comme point d'entrée la classe *HttpClient*. Avant toute chose, il conviendra donc de créer un objet de type *HttpClient* sur lequel nous pourrons travailler pour tester les différentes fonctionnalités offertes par l'API. La création d'un objet *HttpClient* peut se faire de deux manières différentes. La première, qui est la plus simple, permet de créer une instance de *HttpClient* configurée via le paramétrage par défaut dont la version HTTP utilisée de manière sous-jacente est positionnée à HTTP/2 :

```
HttpClient httpClient = HttpClient.newHttpClient();
```

Pour des besoins plus spécifiques, il demeure possible de configurer directement un objet *HttpClient* sur mesure via la méthode statique *newBuilder()* qui est une mise en pratique du célèbre design pattern Builder. Dans l'exemple ci-dessous, on choisit de créer une instance de *HttpClient* s'appuyant sur HTTP/2 et fournissant une authentification basique :

```
HttpClient httpClient = HttpClient.newBuilder()
    .version(HttpClient.Version.HTTP_2)
    .authenticator(new Authenticator() {
      @Override
      protected PasswordAuthentication getPasswordAuthentication() {
        return new PasswordAuthentication("ssaurel", "programmez!".toCharArray());
      }
    })
    .build();
```

Requête GET synchrone

La création de requêtes se fait via l'objet *HttpRequest* de l'API. Là encore, le design pattern Builder a été mis à profit afin de fournir une API lisible et agréable à l'emploi pour le développeur. Nous allons donc pouvoir préciser l'URI à requêter via une méthode *uri()*, puis préciser le type de la requête en appelant la méthode *GET()* dans le cas présent. Une fois la requête configurée selon nos souhaits, l'instance d'objet *HttpRequest* est récupérée en appelant la méthode *build()*.

L'exécution de la requête de manière synchrone se fait ensuite simplement en passant l'objet *HttpRequest* créé précédemment en paramètre de la méthode *send* de l'objet *HttpClient* créé précédemment. La méthode *send* prend un second paramètre en entrée permettant de préciser le format souhaité pour la réponse. Ici, on souhaite obtenir une réponse au format texte ce qui donne le code suivant :

```
HttpClient client = HttpClient.newHttpClient();
HttpRequest request = HttpRequest.newBuilder()
        .uri(new URI("https://www.ssaurel.com/myfile.txt"))
        .GET()
        .build();

HttpResponse<String> response = client.send(request,
                    HttpResponse.BodyHandler.asString());
```

La réponse de notre requête GET synchrone est modélisée via l'objet *HttpResponse*. Il est possible de vérifier le code retour de la requête HTTP/2 en appelant la méthode *statusCode()* et enfin d'accéder au contenu récupéré via la méthode *body()*.

Requête GET asynchrone

Réaliser une requête GET en mode asynchrone est similaire à ce que nous venons de faire à ceci près que c'est la méthode *sendAsync()* de l'objet *HttpClient* que nous allons appeler ici. En outre, la réponse de la requête sera représentée au sein d'un objet *CompletableFuture* dont nous pourrons appeler la méthode *get()* pour préciser le temps d'attente que nous souhaitons fixer avant de vérifier ce que la requête a donné comme résultat. On obtient ainsi le morceau de code suivant :

```
HttpClient client = HttpClient.newHttpClient();
HttpRequest request = HttpRequest.newBuilder()
        .uri(new URI("https://www.ssaurel.com/myfile.txt"))
        .GET()
        .build();

CompletableFuture<HttpResponse<String>> response = client.sendAsync(request,
        HttpResponse.BodyHandler.asString());
HttpResponse<String> actualResponse = response.get(1000, TimeUnit.MINUTES);
System.out.println("Code retour : " + actualResponse.statusCode());
```

Requêtes POST simples

La construction d'une requête POST simple se fait toujours au travers d'un objet *HttpRequest* sur lequel on va appeler la méthode *POST()* pour typer la requête correctement. L'argument de cette dernière méthode permettant de définir les variables que l'on souhaite envoyer côté serveur en mode POST. Il est également possible de préciser au niveau de la requête que l'on souhaite utiliser le protocole HTTP/1.1 même si l'objet *HttpClient* créé a été configuré en HTTP/2. Une requête POST basique aura donc l'allure suivante :

```
HttpClient client = HttpClient.newHttpClient();
HttpRequest request = HttpRequest.newBuilder()
        .uri(new URI("https://www.ssaurel.com/post")).POST(
          HttpRequest.BodyProcessor.fromString("nom=SAUREL,prenom=SYLVAIN"))
          .version(HttpClient.Version.HTTP_1_1)
          .build();

HttpResponse<String> response = client.send(request,
                        HttpResponse.BodyHandler.asString());
System.out.println("Réponse : " + response.body());
```

Envoyer un tableau de bytes se révèle également un jeu d'enfant avec la nouvelle API proposée par Java 9. Pour ce faire, il suffira d'encapsuler le tableau de bytes au sein d'un objet de type *HttpRequest.BodyProcessor* via un appel à la méthode statique *HttpRequest.BodyProcessor.fromByteArray()*. Une requête POST envoyant un tableau de bytes pourra être codée comme suit :

```
HttpClient client = HttpClient.newHttpClient();
byte[] body = "HTTP 2 avec Java 9 pour Programmez".getBytes();

HttpRequest request = HttpRequest.newBuilder()
          .uri(new URI("https://www.ssaurel.com/post")).POST(
              HttpRequest.BodyProcessor.fromByteArray(body))
          .version(HttpClient.Version.HTTP_1_1)
          .build();

HttpResponse<String> response = client.send(request,
                        HttpResponse.BodyHandler.asString());
System.out.println("Code Retour : " + response.statusCode());
System.out.println("Réponse : " + response.body());
```

Upload d'un fichier via une requête POST

L'upload d'un fichier du client vers le serveur est un cas d'utilisation plutôt fréquent dans les applications d'entreprises. La nouvelle API mise à disposition par Java 9 facilite là encore grandement le travail des développeurs. En effet, la seule différence avec les exemples de requêtes POST précédent va se situer au niveau du paramètre de type *HttpRequest.BodyProcessor* passé en entrée de la méthode *POST()* du builder de l'objet *HttpRequest*. Dans le but de préciser que l'on souhaite envoyer un fichier, on créera un objet *BodyProcessor* via sa méthode statique *fromFile()* avec en paramètre le chemin du fichier sur le poste client :

```
HttpClient client = HttpClient.newHttpClient();
File file = new File("/usr/ssaurel/files/myfile.txt");
HttpRequest request = HttpRequest.newBuilder()
          .uri(new URI("https://www.ssaurel.com/post")).POST(
              HttpRequest.BodyProcessor.fromFile(file.toPath()))
          .version(HttpClient.Version.HTTP_1_1)
          .build();

HttpResponse<String> response = client.send(request,
                        HttpResponse.BodyHandler.asString());
System.out.println(response.body());
```

Authentification basique

La nouvelle API client HTTP/2 offre également une gestion simplifiée de l'authentification basique à un service distant. Pour notre exemple, nous allons nous appuyer sur le service basic-auth du site http://httpbin.org qui permet de réaliser rapidement des tests de requêtes HTTP. Comme nous avons pu le voir au début de cet article, nous allons créer une instance d'objet *HttpClient* configurée pour nos besoins bien spécifiques. Nous préciserons ainsi que nous souhaitons nous appuyer sur le protocole HTTP/1.1 pour échanger avec le serveur http://httpbin.org mais également que nous fournissons une implémentation de l'interface *Authenticator* renvoyant notre couple utilisateur/mot de passe.
Une fois cette instance d'objet *HttpClient* obtenue, il ne nous reste plus qu'à construire une requête HTTP de type GET sur l'URI du service sur lequel on souhaite s'authentifier :

```
HttpClient client = HttpClient.newBuilder()
          .version(HttpClient.Version.HTTP_1_1)
          .authenticator(new Authenticator() {
            @Override
            protected PasswordAuthentication getPasswordAuthentication() {
              return new PasswordAuthentication("user", "passwd".toCharArray());
            }
          })
          .build();

HttpRequest request = HttpRequest.newBuilder()
          .uri(new URI("http://httpbin.org/basic-auth/user/passwd"))
          .GET()
          .build();

HttpResponse<String> response = client.send(request,
                        HttpResponse.BodyHandler.asString());
System.out.println("Code retour : " + response.statusCode());
```

Définition des Headers

Une API visant à simplifier l'usage des appels HTTP en Java ne serait pas complète sans possibilité de définir les Headers envoyés lors d'une requête. Bien entendu, les architectes en charge du client HTTP/2 pour Java 9 ont pris en compte cette problématique et il est possible de définir des Headers spécifiques lors la configuration de l'objet *HttpRequest* en recourant aux méthodes *header()* et *setHeader()* :

```
HttpClient client = HttpClient.newHttpClient();
HttpRequest request = HttpRequest.newBuilder()
          .uri(new URI("http://httpbin.org/headers"))
          .version(HttpClient.Version.HTTP_1_1)
          .GET()
          .header("mode","simple")
          .header("type", "txt")
          .setHeader("max", "30")
          .build();

HttpResponse<String> response = client.send(request,
                        HttpResponse.BodyHandler.asString());
```

```
System.out.println("Code retour : " + response.statusCode());
System.out.println("Body du retour : " + response.body());
```

Paramétrage du Timeout pour une Requête

Pouvoir définir un timeout spécifique pour une requête est également une possibilité intéressante offerte par la nouvelle API. Pour tester ce cas d'usage bien particulier, nous allons nous appuyer sur le service delay du site http://httpbin.org . Lors du paramétrage de la requête nous configurons le timeout via la méthode éponyme *timeout()* prenant en entrée le timeout que nous souhaitons appliquer. Si la réponse n'est pas reçue dans le laps de temps que nous avons défini, une exception *HttpTimeoutException* sera alors lancée :

```
HttpClient client = HttpClient.newHttpClient();
int delay = 10; // en secondes

HttpRequest request = HttpRequest.newBuilder()
        .uri(new URI("http://httpbin.org/delay/" + delay))
        .version(HttpClient.Version.HTTP_1_1)
        .GET()
        .timeout(Duration.ofSeconds(delay - 5))
        .build();

try {
  HttpResponse<String> response = client.send(request,
                        HttpResponse.BodyHandler.asString());
} catch (HttpTimeoutExcepion exc) {
  System.out.println("Timeout dépassé pour la requête");
}
```

Mise en place d'un Proxy

La mise en place d'un proxy se fait au niveau de la création de l'instance d'objet HttpClient à l'aide d'un objet ProxySelector. Dans l'exemple qui suit, nous choisissons d'utiliser le Proxy défini par défaut sur le système hôte en appelant la méthode statique getDefault() de ProxySelector :

```
HttpClient client = HttpClient.newBuilder()
        .proxy(ProxySelector.getDefault())
        .build();
```

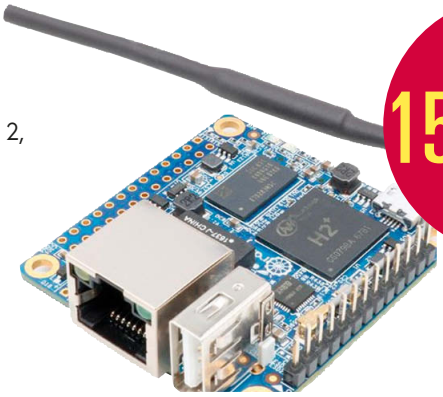
Conclusion

Cet article aura permis de mettre en évidence les différents cas d'usage de la nouvelle API Client HTTP/2 introduite par Java 9. Nous avons ainsi pu voir comment créer des requêtes GET en mode synchrone et asynchrone, comment créer des requêtes POST plus ou moins complexes, mais également comment configurer les Headers d'une requête HTTP ou encore comment configurer plus finement l'objet *HttpClient* avec la mise en place d'une authentification basique. Plus moderne et plus adaptée au Web actuel que l'API tournant autour de l'objet *UrlConnection*, l'API HTTP/2 offre également un support du protocole HTTP/1.1. Elle s'imposera très rapidement auprès des développeurs tant pour les possibilités nouvelles apportées par le protocole HTTP/2 que par le gain de productivité offert par l'API. •

Boutique matérielle

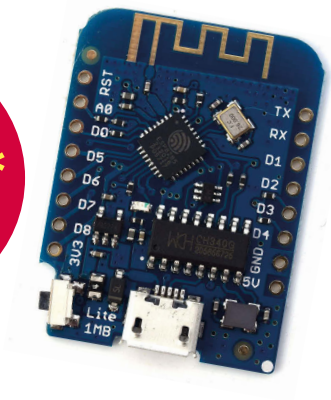
Orange Pi Zero / version 256 Mo

256 Mo de ram, Ethernet 10/100, Wifi + antenne WiFi, 26 pins I/O, USB 2, processeur ARM 1,2 Ghz, carte livrée sans système.
Alternative à la Raspberry Pi Zero



15,99 €*

7,99 €*



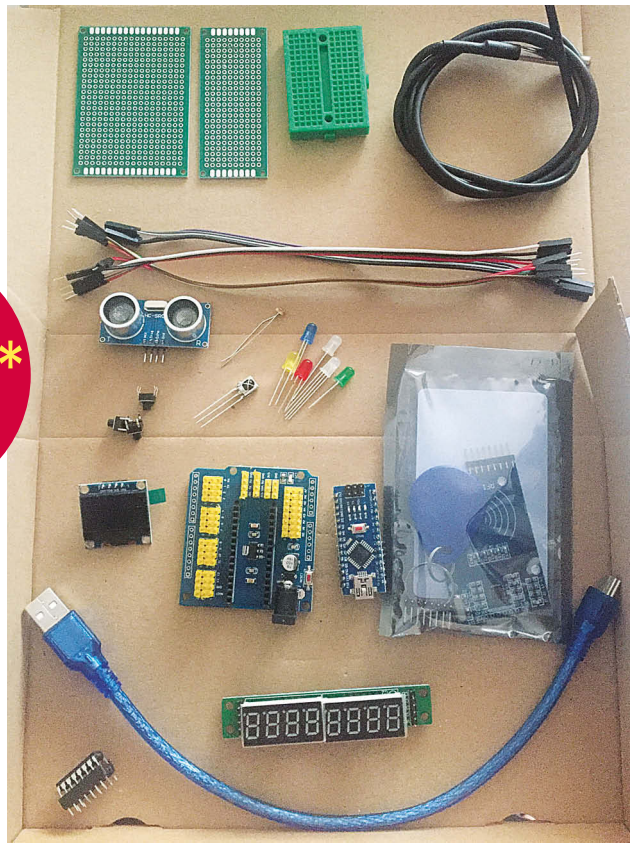
Wemos D1 Mini Lite (ESP8266)

1 mo de stockage flash, WiFi, 16 pins I/O, 10 grammes, compatible Arduino

Pack Maker 12/2017 : Arduino Nano + composants

- Arduino Nano + câble USB
- Carte d'extension I/O
- Mini planche à pain
- Lecteur RFID
- Ecran OLED 0,96 pouce
- Double afficheur LED
- Capteur étanche DS18B20
- 2 plaques PCB
- Capteur ultra-son HC-SR4
- 6 LED
- 1 Capteur de luminosité
- 3 mini-boutons
- 1 récepteur infra-rouge
- 1 contrôleur SN74HC595N : registre à décalage, série vers parallèle, 8 bit, idéal pour horloge et synchronisation

34,99 €**



** Port inclus. tarif France uniquement.*
*** Tarif France uniquement. Port en sus : + 5 €*

Commandez directement sur notre boutique en ligne : www.programmez.com

Ces offres peuvent s'arrêter à tout moment. Quantité très limitée.

Avertissements :

Les composants sont livrés sans documentation.
Programmez!, Nefer-IT, la rédaction, ne peuvent être tenus responsables des détériorations occasionnées par l'utilisation des composants, ni d'un mauvais usage. Matériel électronique : respecter les règles de sécurité et les recommandations de montage des constructeurs. L'utilisation de ces matériels est sous la seule responsabilité de l'acheteur.

Olivier Philippot
CTO chez Greenspector
Il aime les technologies, mais surtout les optimiser. En particulier sur Android et l'embarqué. Il est fortement engagé dans les actions de sensibilisation à l'efficience et l'éco-conception logicielle. Speaker a Devoxx, Android Maker, Breizhcamp...

Améliorez votre référencement SEO avec l'écoconception en hackant les budgets Google Crawl

Partie 2

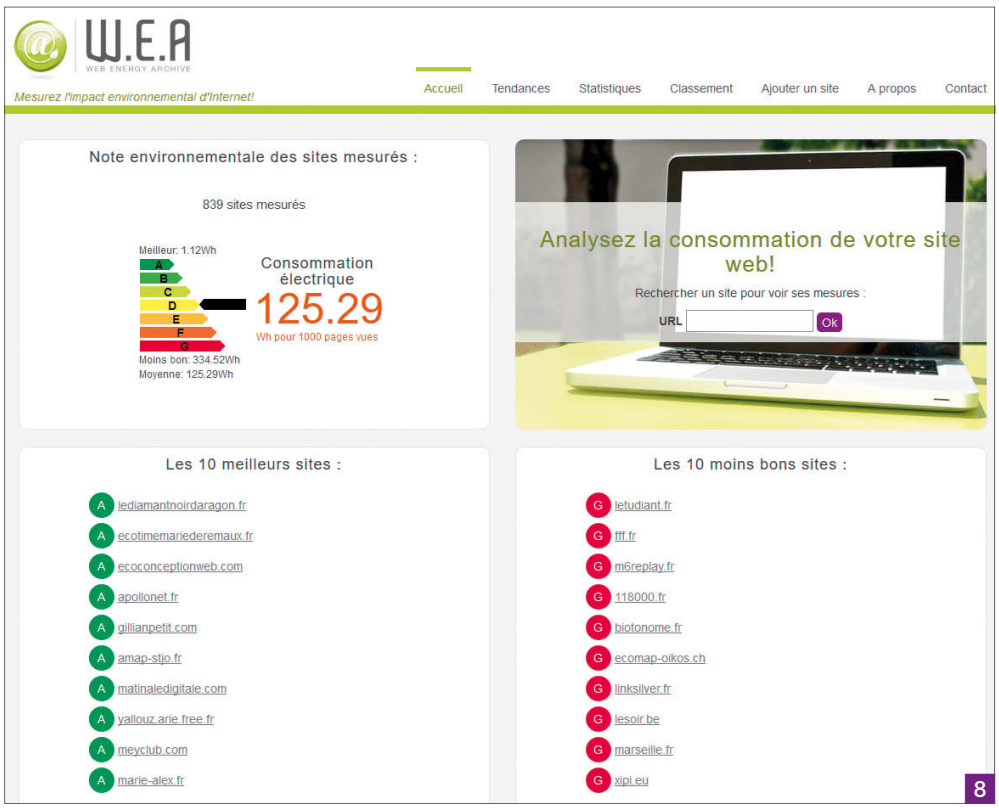
Le référencement SEO est un domaine critique pour le succès des sites et d'applications web. En effet, sans un bon référencement, pas ou peu d'utilisateurs, et donc une solution qui ne sert pas. À l'opposé, les algorithmes de référencement sont peu transparents et les articles de blog fourmillent de préconisations pour améliorer son classement. La performance est une des pistes pour améliorer le référencement SEO. Mais est-ce que c'est bien vrai ?

Aller plus loin avec l'écoconception

La limite de la performance pure

Se focaliser uniquement sur la performance serait une erreur. En effet, les bots ne sont pas des utilisateurs comme les autres. Ce sont des machines, pas des humains. La performance web se focalise sur le ressenti utilisateur (affichage le plus rapide). Or les bots voient plus loin que les humains, ils doivent en effet lire tous les éléments (CSS, JS...). Par exemple, si vous vous focalisez sur la performance d'affichage, vous allez potentiellement appliquer la règle qui conseille de déférer certains codes JavaScript en fin de chargement. De cette manière la page apparaît rapidement et les traitements continuent ensuite. Or comme les bots ont pour objectif de crawler tous les éléments de la page, le fait que les scripts soient déférés ne changera rien pour eux.

Il est donc nécessaire d'optimiser tous les éléments, y compris ceux qui se chargeront une fois la page affichée. On fera particulièrement attention à l'usage du JavaScript. Google Bot peut le parser, mais avec des efforts. On utilisera donc chaque technologie avec parcimonie. Par exemple AJAX a pour but de créer des interactions dynamiques avec la page : oui pour les formulaires ou chargements de widgets... mais non si c'est uniquement pour un chargement de contenu sur un OnClick. Faciliter le travail du GoogleBot revient à travailler sur tous les éléments du site. Moins il y a d'éléments, plus c'est facile à faire. Donc : Keep It Simple !...



8

L'écoconception à la rescousse du crawl

L'écoconception a pour objectif principal de réduire l'impact environnemental des logiciels. Techniquement, cela se traduit par la notion d'efficience : il faut répondre au besoin de l'utilisateur, en respectant les contraintes de performance, mais avec une consommation de ressources et d'énergie qui soit la plus faible possible. La consommation d'énergie causée par les logiciels (dont votre site web) est donc la pierre angulaire de la démarche.

Comme vous l'avez compris depuis le début de l'article, les bots ont un budget issu d'une capacité limitée en temps, mais aussi une facture d'énergie à limiter pour les datacenters de Google. Tiens ? Nous retrouvons ici le but de l'écoconception des logiciels : limiter cette consommation d'énergie. Et pour cela, une approche basée sur la performance ne convient pas : en cherchant à améliorer la performance sans maintenir sous contrôle la consommation de ressources induites, vous risquez fort d'obtenir un effet inverse à celui que vous recherchez.

Alors, quelles sont les bonnes pratiques ? Vous pouvez parcourir le blog de Greenspector et vous en découvrirez :-) Mais je dirais que peu importe les bonnes

pratiques, c'est le résultat sur la consommation de ressources qui compte. Est-ce que les Progressive web App (PWA) sont bonnes pour le crawling ? Est-ce que le lazy loading est bon ? Peu importe. Si, au final, votre site consomme peu de ressources, et qu'il est facile à "afficher" par le bot, c'est bon.

La mesure d'énergie pour contrôler la crawlabilité

Une seule métrique englobe toutes les consommations de ressources : l'énergie. En effet la charge CPU, les requêtes réseaux, les traitements graphiques... tout cela va déboucher sur une consommation d'énergie par la machine. En mesurant et contrôlant la consommation d'énergie, vous allez pouvoir contrôler le coût global de votre site. Voici par exemple une mesure de consommation d'énergie du site Nantes transition énergétique . On voit 3 étapes : le chargement du site, l'idle au premier plan, et l'idle navigateur en tâche de fond. De nombreuses bonnes pratiques d'écoconception ont été appliquées. Malgré cela, un slider tourne et la consommation en idle reste importante, avec une conso totale de 80 mWh. 9

On le voit ici, la consommation d'énergie est le juge de paix de la consommation de ressources. Réduire la consommation d'énergie va vous permettre de réduire l'effort du GoogleBot. Le budget énergétique idéal pour le chargement d'une page est de 15 mWh : c'est celui qui permet de décrocher le niveau "Or" du Green Code Label. Pour rappel ici, malgré le respect de bonnes pratiques, la consommation était de 80 mWh. Après analyse et discussions entre les développeurs et le maître d'ouvrage, le slide a été remplacé par un affichage aléatoire et statique d'images : la consommation est retombée sous les 15mWh !

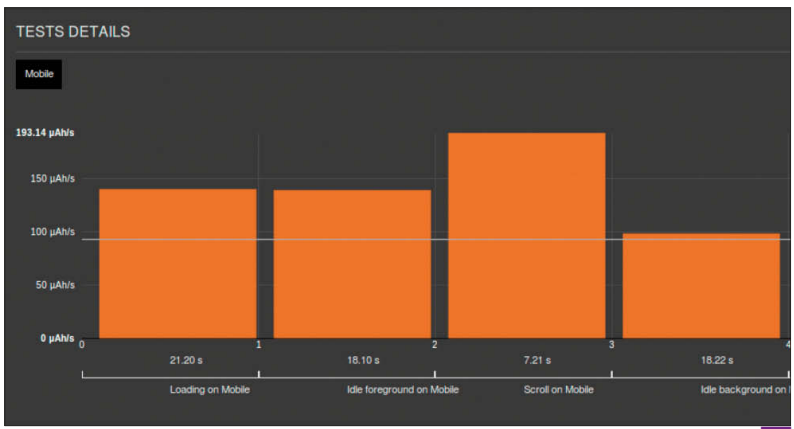
En attendant de vous faire labelliser :-) vous pouvez mesurer gratuitement la consommation de votre site sur web Energy Archive, un projet associatif auquel nous sommes fiers d'avoir contribué.

L'écoconception des fonctionnalités

Améliorer la consommation et l'efficience de votre page, c'est bien, mais cela risque de ne pas suffire. L'étape suivante est d'optimiser aussi le parcours de l'utilisateur. Le bot est un utilisateur ayant un budget ressource limité et devant parcourir le maximum d'URL de votre site avec ce budget. Si votre site est complexe à parcourir (trop d'URL, une profondeur trop importante...) le moteur ne parcourra pas tout le site, ou alors en plusieurs fois.

Il est donc nécessaire de rendre le parcours utilisateur plus simple. Cela revient encore à appliquer des principes d'écoconception. Quelles sont les fonctionnalités dont l'utilisateur a vraiment besoin ? Si vous intégrez des fonctionnalités inutiles, l'utilisateur et le bot vont gaspiller des ressources au détriment des éléments vraiment importants. Outils d'analytiques trop nombreux, animations dans tous les sens... demandez-vous si vous avez vraiment besoin de ces fonctionnalités, car le bot devra les analyser. Et cela se fera sûrement au dépit de votre contenu qui, lui, est important. Pour l'utilisateur, c'est pareil : si obtenir une information ou un service demande un nombre de clics trop important, vous risquez de le décourager. Une architecture de parcours efficiente va permettre d'aider le bot et aussi l'utilisateur. Aidez-vous du sitemap pour analyser la profondeur du site.

Enfin, l'exercice d'écoconception des pages ne sera pas efficace si vous avez un site complexe. Il est possible, par exemple, qu'avec des urls dynamiques un nombre important d'urls soit détecté par Google. Vous aurez alors des urls en double, du contenu inutile... Un nettoyage de l'architecture du site sera peut-être nécessaire ?



9

ÉCOCONCEPTION DES LOGICIELS

L'enjeu de l'écoconception est d'intégrer les enjeux environnementaux dès la phase de conception des produits en se basant sur la prise en compte de l'ensemble des étapes du cycle de vie des produits. Même s'ils sont considérés comme immatériels, les logiciels ont besoin d'équipements informatiques pour fonctionner. Or l'impact environnemental de ces équipements ne cesse de grossir, et leur empreinte écologique se concentre sur les étapes de fabrication et de fin de vie. Pour rendre le logiciel plus respectueux de l'environnement, il est donc primordial d'allonger la durée d'utilisation des équipements informatiques.

Or la couche logicielle est un facteur d'obsolescence non négligeable, car les ressources matérielles nécessaires (quantité de mémoire vive, puissance du processeur, espace disque) ne cessent d'augmenter.

L'écoconception appliquée aux logiciels consiste donc à optimiser les logiciels dès leur conception pour :

- diminuer la consommation énergétique du logiciel pendant sa phase d'utilisation (et donc les émissions de gaz à effet de serre liées) ;
- améliorer la performance des applications et sites web, l'autonomie des appareils mobiles pour une meilleure expérience utilisateur ;
- allonger la durée de vie des équipements informatiques (et donc diminuer leur impact environnemental).

L'écoconception logicielle consiste à appliquer des bonnes pratiques pour diminuer la consommation en ressources matérielles tout en maintenant la performance et en répondant au besoin de l'utilisateur.

Anticiper les algorithmes Mobile First de Google

Prise en compte des nouvelles plateformes

Google veut améliorer l'expérience de recherche des utilisateurs, il prend donc en compte les derniers usages des utilisateurs. Le nombre de visiteurs mobiles excédant maintenant celui des visiteurs desktop, les algorithmes de Google passent au Mobile First. Le planning n'est pas très clair, mais plusieurs signes annoncent ce changement.

Vous pouvez anticiper la manière dont Google crawle votre site, toujours via la

console Search Crawl, avec l'option Fetch as Mobile. 10

Google propose aussi un site de test pour savoir si votre site est prêt pour le mobile. 11

La prise en compte de ces contraintes Mobile First n'est pas forcément simple pour votre site. Il est probable que cela vous prenne du temps et quelques efforts. Mais c'est le prix à payer si vous voulez avoir un bon référencement dans les prochains mois. En outre, vos visiteurs seront reconnaissants : ils apprécieront que votre site ne décharge plus autant la batterie de leur smartphone.

Pour aller encore plus loin

Reprenons la citation de Google : *Googlebot is designed to be a good citizen of the web.*

En tant que développeur ou propriétaire de site, votre ambition doit être de devenir au moins un "aussi bon citoyen du web" que le bot de Google. Pour cela, projetez-vous dans un usage de votre site sur plateforme mobile, mais aussi dans des conditions de connexion bas débit ou d'appareils un peu plus anciens. En réduisant les consommations de ressources de votre site, non seulement vous ferez plaisir au GoogleBot (qui vous récompensera par un meilleur référencement), mais aussi vous donnerez à toute une partie de la population mondiale (y compris dans les pays "riches") la possibilité d'accéder à votre site. Ne croyez pas que cette approche soit uniquement philanthropique. Rappelons que Facebook et Twitter l'appliquent déjà avec les versions "Light" de leurs applis exemple : Facebook Lite qui utilise 90 % de données en moins et consomment deux fois moins que la version officielle. Un site étant non seulement compatible sur mobile, mais prenant en compte toutes les plateformes mobiles et toutes les vitesses de connexion devrait donc normalement être valorisé dans un futur proche par Google. D'autant plus que, plus le site sera léger, moins Google consommera de ressource. Et comme pour l'utilisateur, la consommation, et surtout l'impact sur l'environnement, devient de plus en plus important :

- Obligation de rédiger des rapports de développement durable ;
- Pression des ONG pour réduire ses impacts, par exemple le rapport Greenpeace.

Il est certain qu'un site efficient et répondant uniquement au besoin de l'utilisateur sera mieux valorisé dans le classement SEO de Google (et des autres).

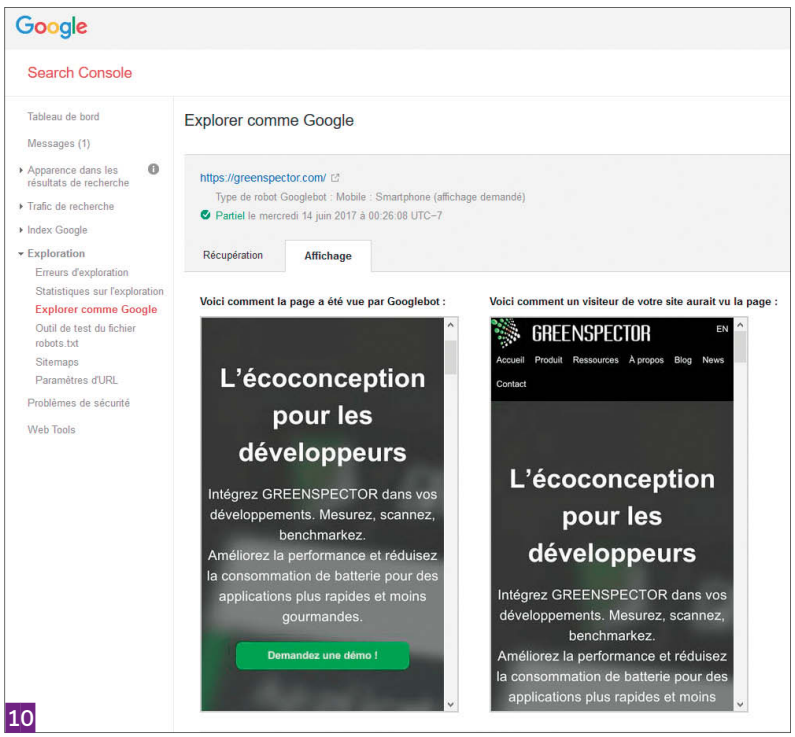
11

Conclusion

Le référencement de votre site web par Google s'appuie en partie sur la notion de crawling : le robot parcourt les pages de votre site. Pour cette opération, il se fixe un budget de temps et de ressources. Si vos pages sont trop lourdes, trop lentes, le crawling ne sera réalisé que partiellement. Votre site sera mal référencé.

Pour éviter cela, commencez par prendre l'habitude d'évaluer la "crawlabilité" de votre site. Puis mettez en place une démarche de progrès. En première approche, regardez ce que vous pouvez obtenir à l'aide des outils de performance classiques. Puis réfléchissez en termes de Mobile First : appliquez les principes de l'écoconception à votre site. Simplifiez le parcours utilisateur, enlevez ou réduisez les fonctions inutiles. Mesurez votre consommation d'énergie et mettez là sous contrôle.

Au prix de ces quelques efforts, vous obtiendrez un site mieux référencé, plus performant, apprécié des utilisateurs. De nombreux sites web ayant cédé à l'obésité (trop de requêtes, des scripts superflus...), il est urgent d'anticiper l'évolution des algorithmes de référencement de Google, qui suivent désormais les mêmes objectifs que les utilisateurs, en demandant plus de performance ET moins de consommation d'énergie : en un mot, de l'efficience. •



10

LE GREEN CODE LABEL : passez à l'action sur votre site !

Green Code Label évalue le niveau d'écoconception d'un site web. Ce projet à l'initiative du Green Code Lab est basé sur un référentiel open source (https://greencodelab.github.io/) de bonnes pratiques d'écoconception web. Ce référentiel couvre plusieurs domaines : code source, contenu, design, hébergement. Une attention particulière est également apportée à la mesure énergétique du site web à l'aide de web Energy Archive (http://webenergyarchive.com/).

Parmi les règles du référentiel, on retrouve à la fois des règles simples et faciles à appliquer (optimisation des images, mise en cache, minification), mais aussi des règles qui amènent à une réflexion plus globale sur les besoins fonctionnels du site et les choix d'architecture, toujours dans une optique de sobriété du numérique. Un audit réalisé par un consultant certifié permet de sanctionner l'éligibilité du site web à l'obtention du label. Selon le niveau d'écoconception du site, trois niveaux de labellisation peuvent être obtenus : or, argent ou bronze.

Dans ce cas, le label peut être affiché sur la page d'accueil du site pour une durée de 2 ans.

Patrick Giry
Software Gardener Arolla

co-auteur
Cyrille Martraire
https://twitter.com/cyriux
CTO d'Arolla

Le dilemme entre code expressif et code générique FP

Utiliser au mieux la plomberie fournie par le langage de programmation ou exprimer au mieux le domaine métier ? C'est le dilemme habituel dans un langage de programmation tel que Java. Voyons cela de plus près au travers de quelques exemples, et comment nous parvenons à résoudre ce problème avec plus ou moins de bonheur selon le langage de programmation choisi.

Type standard et type métier

Prenons l'exemple du prix et de la quantité en e-commerce. Nous souhaitons créer un type `Price` autour d'un *double* et un type `Quantity` autour d'un *int* afin d'exprimer au mieux le métier avec des types. Cela amène aussi la protection du typage pour éviter de passer un `int` à la place d'un autre par erreur.
En Java, il faut donc créer une classe pour chaque :

```
public final class Price {
   private final double value;

   public Price(double value) { this.value = value; }

   ...
}

public final class Quantity {
   private final int value;

   public Quantity(int value) { this.value = value; }

   ...
}
```

Maintenant nous voulons multiplier le prix par la quantité pour calculer le prix total, qui doit être aussi de type `Price` :

```
public final class Price {
  ...
  public Price multiply(double multiplicand) {
    return new Price(this.value * multiplicand);
  }
}

public final class Quantity {
  ...
  public double asDouble() { return value; }
}
```

Nous aimerions pouvoir passer directement une instance de `Quantity` en paramètre, mais cela nous oblige à sortir la valeur primitive :

```
myPrice.multiply(myQuantity.asDouble());
```

Ou alors à modifier la méthode `multiply()` pour accepter le type `Quantity`, ce qui la rend désormais spécifique à ce type, et donc couplée par la même occasion :

```
Price multiply(Quantity quantity) {
  return new Price(this.value * quantity.asDouble());
}
```

Cela dit, d'un point vue métier, nous pouvons admettre qu'il est possible de multiplier un prix par une quantité.
En Haskell, la notion de synonyme ou alias permet de donner un nom plus métier au type :

```
type Price = Double
type Quantity = Int
```

Nous avons un type sur-mesure, mais que nous pouvons aussi utiliser comme un entier quand nous le souhaitons, car il reste aussi un entier :

```
multiply :: Price -> Quantity -> Price
multiply p q = p * (asDouble q)
    where asDouble = fromIntegral
```

Notez qu'on peut appeler la fonction `multiply` avec n'importe quel entier.

```
multiply 4.5 5 `shouldBe` 22.5

multiply 4.5 (6 :: Int) `shouldBe` 27.0
```

Si nous souhaitons faire des types wrapper comme en Java, nous utilisons des `newtype` :

```
newtype Price = Price Double
  deriving (Eq, Show)

newtype Quantity = Quantity Int
  deriving (Show)
```

La fonction `multiply` s'appuie alors sur des "déconstructeurs" pour extraire les valeurs du prix et de la quantité :

```
multiply :: Price -> Quantity -> Price
multiply (Price p) (Quantity q) = Price (p * (asDouble q))
    where asDouble = fromIntegral
```

Une fois le calcul réalisé, nous construisons le prix avec le résultat. Pour appeler la fonction, il faut créer un prix et une quantité :

```
(Price 4.5) `multiply` (Quantity 5) `shouldBe` Price 22.5
```

Là, nous sommes donc vraiment "type-safe". De plus, comme la fonction `multiply` a deux arguments, nous pouvons l'infixer. Il est plus naturel de dire "le prix multiplié par la quantité".
Une autre façon de contourner le problème consistant à être à la fois un type standard et un type sur-mesure métier est la conversion implicite (implicit cast en C++, C# ou Scala). Java supporte une construction similaire, le "unboxing", mais seulement pour les types primitifs.

Monoid standard et monoid métier

Maintenant, nous souhaitons additionner des prix. Nous définissons donc une méthode add() à la class Price :

```
Price add(Price other) {
 return other != null ? new Price(value + other.value) : this;
}
```

Et par confort, j'ajoute le prix zéro, bien utile dans de nombreux cas :

```
public static final Price ZERO = new Price(0.);
```

Nous avons donc de fait un type qui obéit à la structure et aux propriétés d'un Monoid. Pas d'inquiétude! Si vous ne savez pas ce que c'est, ce n'est pas indispensable pour cet article.
Imaginons que nous avons une interface Monoid disponible que nous utilisons déjà un peu partout :

```
public interface Monoid<M extends Monoid<M>> {

  M append(M other);

  M empty();

  default M concat(M... ms) {
    return Stream.of(ms).reduce(empty(), Monoid::append);
  }
}
```

Nous souhaitons alors que Price implémente l'interface Monoid :

```
class Price implements Monoid<Price> {
  ...
  public Price add(Price otherPrice) {
    return append(otherPrice);
  }

  public Price total(Price... prices) {
    return concat(prices);
  }

  @Override
  public Price append(Price other) {
    return other != null ? new Price(value + other.value) : this;
  }

  @Override
  public Price empty() {
    return ZERO;
  }
}
```

Cela nous oblige à définir les méthodes obligatoires **génériques** `append()` et `empty()`, qui renvoient vers les membres **expressifs** du métier, la méthode `add()` et le champ ZERO. Cela va encombrer la classe.
En F# ou Haskell, il est possible là-aussi de définir des **alias de fonctions**. Cela permet d'avoir une fonction nommée selon le domaine métier, tout en étant en même temps l'implémentation d'une fonction définie dans une "interface" sous un autre nom :

```
import Data.Monoid

instance Monoid Price where
  mempty = Price 0
  mappend (Price p1) (Price p2) = Price (p1 + p2)

add :: Price -> Price -> Price
add = mappend

total :: [Price] -> Price
total = mconcat

(Price 10.0) `mappend` (Price 20.0) `shouldBe` Price 30.0

(Price 10.0) `add` (Price 20.0) `shouldBe` Price 30.0

mconcat [Price 12.0, Price 13.0, Price 14.0] `shouldBe` Price 39.0

total [Price 12.0, Price 13.0, Price 14.0] `shouldBe` Price 39.0
```

Predicate standard et predicate métier

Un dernier exemple pour la route, cette fois-ci avec les `Predicate` Java 8 et une astuce !
Nous avons de nombreuses règles de dégressivité sur les quantités, que nous avons modélisées sous forme de critères :

```
public interface QuantityCriteria {
  boolean isSatisfied(Quantity q);
}
```

Le nommage correspond au langage du métier dans notre domaine. Cependant, il est évident que notre critère n'est autre qu'un prédicat, et pour pouvoir utiliser toute la plomberie fournie par le langage de programmation autour des prédicats, il faudrait implémenter l'interface `Predicate` standard :

```
public interface QuantityCriteria extends Predicate {
  boolean isSatisfied(Quantity q);
}
```

Ce qui oblige alors à définir en plus la méthode obligatoire `test()`, un petit encombrement :

```
public interface QuantityCriteria extends Predicate {
  boolean isSatisfied(Quantity q);
  boolean test(Quantity q);
}
```

En outre, il faut que la méthode `test()` renvoie à la méthode `isSatisfied()`. Pour éviter de faire
cela dans chaque implémentation concrète, nous pouvons utiliser une méthode *default* dans notre interface:

```
public interface QuantityCriteria extends Predicate {
  boolean isSatisfied(Quantity q);
  default boolean test(Quantity q) { return isSatisfied(q); }
}
```

Et voilà ! Nous pouvons donc désormais passer n'importe quelle instance de `QuantityCriteria` à toute fonction qui attend un `Predicate` en paramètre. C'est cool!
Le `QuantityCriteria` peut également être implémenté par une lambda, notamment dans la classe `Quantity` pour accéder à sa valeur sans casser l'encapsulation :

```
public final class Quantity {
  ...
  public static QuantityCriteria criteria(IntPredicate predicate) {
    return quantity -> predicate.test(quantity.value);
  }
}
```

Imaginons maintenant que nous voulions composer plusieurs `QuantityCriteria` avec un AND logique :

```
QuantityCriteria over10 = Quantity.criteria(v -> v > 10);
QuantityCriteria below100 = Quantity.criteria(v -> v < 100);
Predicate between10And100 = over10.and(below100);
```

Notez que la méthode `and` intégrée retourne un simple `Predicate`, et non une `QuantityCriteria`. Dommage ! Pourtant, nous souhaiterions utiliser les critères composés, de type `Predicate<Quantity>`, dans la méthode `isEligible` qui attend un `QuantityCriteria`, idéalement :

```
boolean eligible = myPurchaseHistory.isEligible(between10And100);
```

Eh bien, c'est impossible ! En contournement, nous pouvons enrichir `QuantityCriteria` d'une méthode `and` d'adaptation :

```
public interface QuantityCriteria extends Predicate {
  boolean isSatisfied(Quantity q);

  default boolean test(Quantity q) { return isSatisfied(q); }

  default QuantityCriteria and(QuantityCriteria other) {
    return quantity -> Predicate.super.and(other).test(quantity);
  }
}
```

Le problème avec cette approche est qu'il faudra ensuite redéfinir toutes les autres méthodes autour des `Predicate`, par exemple : `or`, `all`, `any`, `not`.
Mais tout n'est pas perdu ! Par exemple, nous pouvons utiliser une méthode référence (équivalente à une lambda) :

```
myHistoryPurchase.isEligible(between10And100::test);
```

Ou encore enrichir `QuantityCriteria` avec un adapter :

```
public interface QuantityCriteria extends Predicate {
  boolean isSatisfied(Quantity q);

  default boolean test(Quantity q) {
    return isSatisfied(q);
  }

  static QuantityCriteria from(Predicate predicate) {
    return predicate::test;
  }
}
```

Ce qui permet ensuite de convertir à la volée un Predicate générique dans son équivalent typé sur-mesure métier :

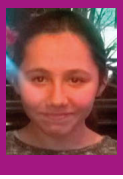
```
myHistoryPurchase.isEligible(QuantityCriteria.from(between10And100));
```

Notez en bonus que `QuantityCriteria` n'a qu'une seule méthode abstraite, et est donc une `FunctionalInterface`. Il est donc possible de passer une instance de `QuantityCriteria` partout où une lambda de même signature est attendue :

```
myStream.filter(myQuantityCritera);
```

Dans les cercles d'amateurs de programmation fonctionnelle, il existe bien souvent un biais très fort pour le générique, au détriment parfois de l'expressivité du langage et des concepts métiers. Ce n'est pas une fatalité.
Et si ça devenait un jeu, d'exprimer le métier le plus littéralement possible, dans ses termes propres, tout en gardant tous les avantages propres aux langages et à leurs écosystèmes disponibles ?
Cet article a été écrit en collaboration avec Cyrille Martraire.
Si vous souhaitez retrouver les sources de cet article, elles sont disponibles sur mon Github :
https://github.com/PatrickGIRY/express_code_and_fp_generic_code.git •

Christophe PICHAUD
Consultant sur les technologies Microsoft
christophepichaud@hotmail.com |
www.windowscpp.net



Pour Lisa.

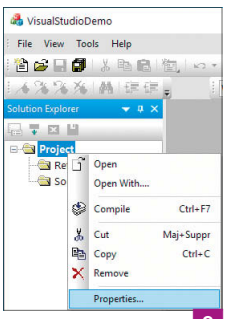
Développer un IDE en C++

Partie 2

Construire un IDE est une aventure, et cela fait partie de ces applications dans lesquelles vous passez du temps pour valider l'ergonomie et découvrir les fonctionnalités qu'il vous manque. En un mot, je développe de l'intérieur. La première chose qu'il me manque c'est l'IntelliSense mais avec le temps je m'en passe. Comme dans Visual Studio je tape entièrement les mots, cela ne me manque pas trop. Après il y a un assistant comme MFC Class Wizard. Là, je m'appuie sur l'existant de mon code, car pour ajouter une commande et définir une fonction membre dans une classe, je peux le faire tout seul. Bref, l'analogie est très simple : au début c'est dur et puis avec le temps, on prend ses marques et on laisse parler l'expérience. Le mois dernier, je vous parlais d'un projet de réaliser un IDE léger.
Ce mois-ci, je vous propose de découvrir les fonctionnalités qui ont été apportées au projet. Le projet est accessible sur GitHub ; si vous voulez jeter un coup d'œil aux sources, c'est le moment !

Montre-moi ton code

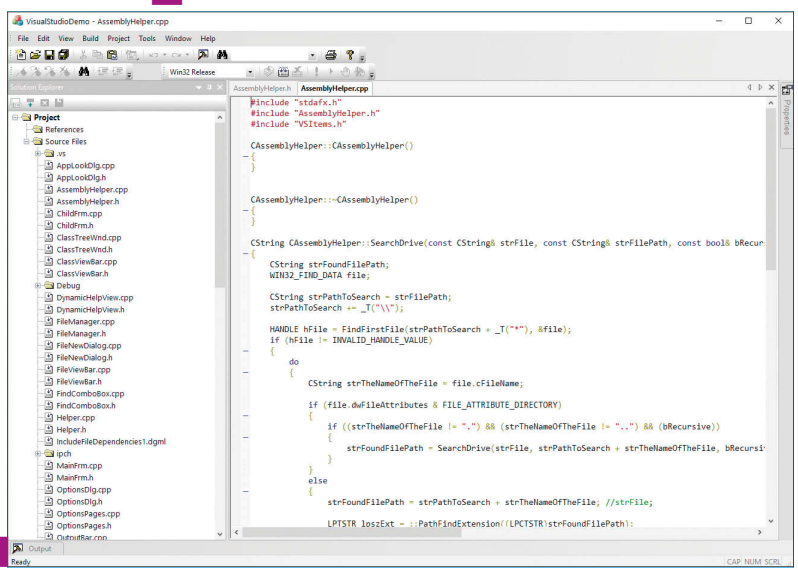
Comme le support de cet article est avant tout un développement concret, on a décidé avec François Tonic (le redac'chef) de mettre le code sur GitHub. Voici : l'URL : https://github.com/ChristophePichaud/VSDemo
Ce développement peut servir de base à la construction d'une application MDI qui implémente des panneaux accrochables (Docking Pane) et des toolbars flottantes. Vous verrez l'impressionnante fonctionnalité Full-Screen plus loin dans l'article.



2

Downloader un package binaire

Si vous n'avez pas de Visual Studio 2015 pour compiler les sources, récupérez une archive zip sur http://www.windowscpp.com/Appz/VSDemo.zip


1

Architecture de code

Pour comprendre comment est architecturé le code, il faut revenir aux fondamentaux. Cette application MFC est MDI donc cela veut dire que plusieurs fenêtres filles peuvent s'afficher sur la MainFrame. Si vous avez lu l'article du mois dernier, vous savez que la fenêtre qui contient le code source est implémentée par la librairie Open-Source Scintilla. Maintenant que l'éditeur de code est fait avec une classe dérivée de CView, on peut se concentrer sur les autres fonctionnalités.

Editer du code librement

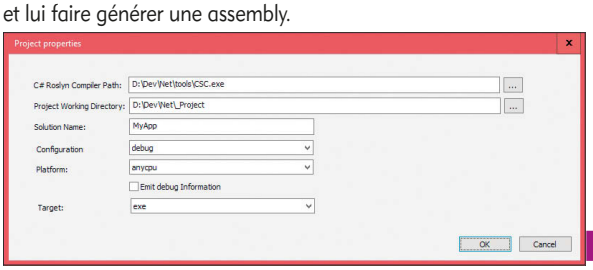
La fonctionnalité « Open Folder » permet de visualiser tous les fichiers d'un répertoire et ainsi de pouvoir les éditer. C'est avec cette fonctionnalité que je fais du dog-fooding. Dans l'image ci-dessous, j'ai fait « Open Folder » de « D:\Dev\GitHub\VSDemo\VisualStudio Demo » qui est l'endroit où sont les sources de l'application. 1

La notion de projet et de solution

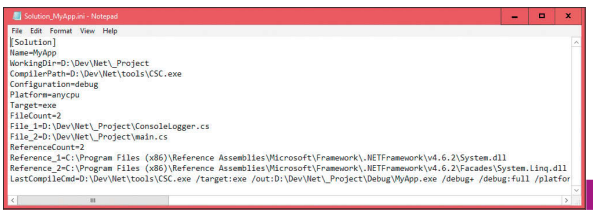
Dans VS, il peut y avoir plusieurs projets dans une solution. Ici, il n'y a qu'un projet principal actif et la solution est un fichier INI dans lequel je renseigne tous les paramètres du projet. Pour obtenir les propriétés du projet, il faut utiliser le pop-menu accessible en faisant bouton-droit sur l'item « Project » et choisir « Properties »: 2
Une boîte de dialogue permet de préciser les grandes caractéristiques du projet à savoir :

- Le chemin vers le compilateur Roslyn ;
- Le répertoire de travail du projet ;
- Le nom de la solution ;
- La configuration (Debug, Release) ;
- La plateforme (x86, x64, AnyCpu) ;
- La target (exe, dll). 3

Voici le fichier solution INI : 4
Le fichier contient explicitement le nom des fichiers du projet ainsi que des assemblies qui font office de références. A partir de ces informations, j'ai tout à disposition pour lancer le compilateur Roslyn et lui faire générer une assembly.



3



4

Rappels sur l'application

Le projet en l'état ressemble à ça… On y trouve un menu « File » riche : 5

« New » permet de créer un fichier. « Add Existing » permet d'ajouter un fichier au projet. « Open Solution » permet d'ouvrir une solution. « Open Folder » permet de visualiser une arborescence de fichiers et de répertoires. C'est ma fonctionnalité préférée.

Par défaut, il y a toujours un projet. 6

Un projet contient des fichiers et des références à des assemblies .NET, qu'elles proviennent du GAC ou du Framework 4.6.2. Pour ajouter un fichier au projet, il faut sélectionner « New ». A partir de là, une boîte propose de créer le fichier : 7

Pour gérer les références, il faut faire « Properties » sur l'item References : 8 9

Cette boîte permet de choisir des références, soit depuis le répertoire du Framework 4.6.2, soit depuis le GAC, soit via la boîte de dialogue commune Ouvrir pour sélectionner un fichier.

A partir de là, on dispose techniquement de toutes les composantes nécessaires à la création d'un projet et sa compilation avec les informations minimales. Voici le code qui permet de localiser les assemblies du framework ou dans le GAC :

```
CString CAssemblyHelper::GetWindowsDirectory()
{
  TCHAR lpszWinDir[MAX_PATH];
  ::GetWindowsDirectory(lpszWinDir, MAX_PATH);
  CString str = lpszWinDir;
  return str;
}

CString CAssemblyHelper::GetFrameworkPath()
{
  TCHAR lpszBuffer[255];
  DWORD dw = 255;

  ::GetEnvironmentVariable(_T("ProgramFiles(x86)"), lpszBuffer, dw);
  CString strWinDir = lpszBuffer;
  CString str;
  str.Format(_T("%s\\Reference Assemblies\\Microsoft\\Framework\\.NETFramework
\\v4.6.2"), strWinDir);
  return str;
}

CString CAssemblyHelper::GetGACPath()
{
  CString strWinDir = CAssemblyHelper::GetWindowsDirectory();
  CString str;
  str.Format(_T("%s\\assembly"), strWinDir);
  return str;
}
```

Vous allez me dire, « ouaip, le Framework 4.6.2 est codé en dur ! ». Oui je sais mais ça va changer. Je donnerais la possibilité de choisir le Framework .NET dans une combo-box. Bref, comme vous le savez, le développement ça prend du temps et l'article doit être déposé très rapidement compte tenu des vacances de Noël donc certaines parties sont hard-codées. Vilain !

Les fonctionnalités de Recherche

CTRL+F lance la boite de dialogue de Recherche : 10

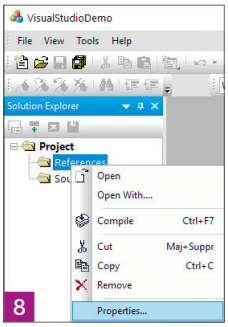
Elle fonctionne en *modeless* et surligne le code en arrière-plan. Cette fonctionnalité est très pratique.

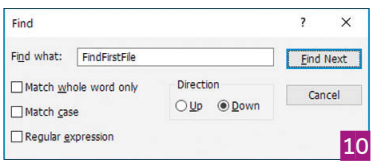
La fonctionnalité « Full Screen »

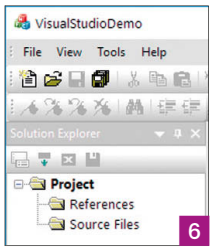
Ajoutez ce bouton via le *customize* de la toolbar ; fonctionnalité disponible dans Build : 11

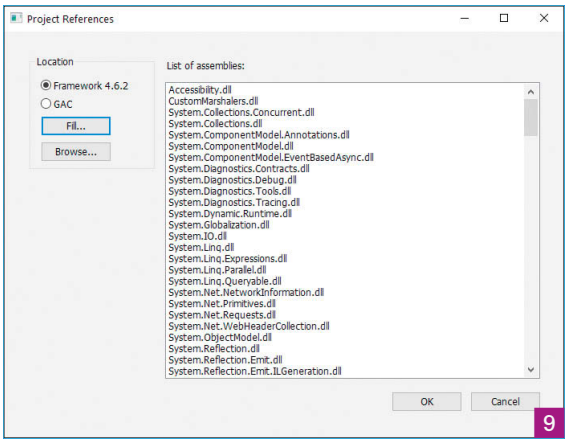
Cette fonctionnalité permet d'avoir un large aperçu du code. C'est assez confortable. 12

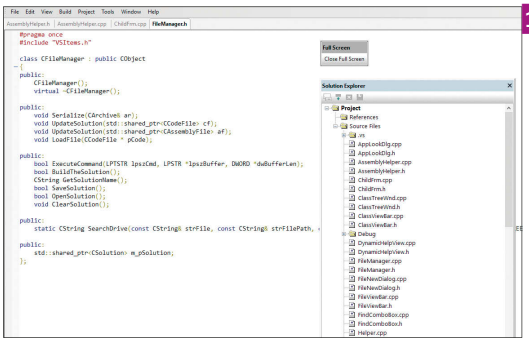
De plus, en demandant l'affichage du DockablePane « Solution Explorer » que l'on laisse en flottant, c'est beau !











Les fonctionnalités qui manquent

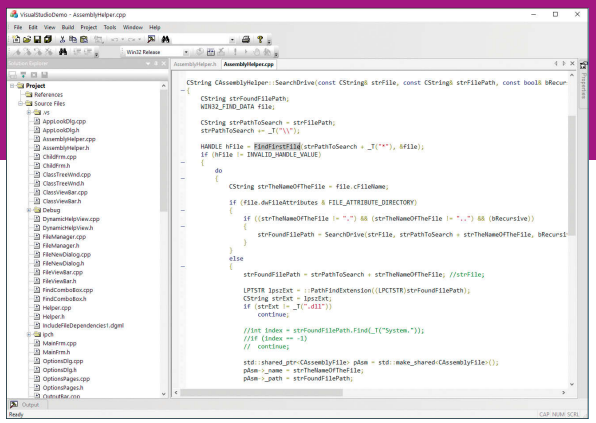
Il faut que je gère :

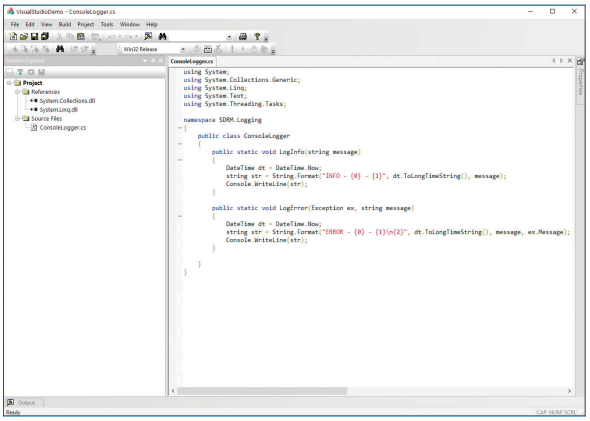
- La toolbar avec les combo-box Recherche et Target ;
- Le bouton Run qui lance le module si c'est un exe ;
- Le clean de la solution ;
- La gestion de plusieurs projets dans la solution ;
- Le passage du fichier de solution a peut-être autre chose que les fichiers INI ;
- La fonctionnalité « Find in Files ».

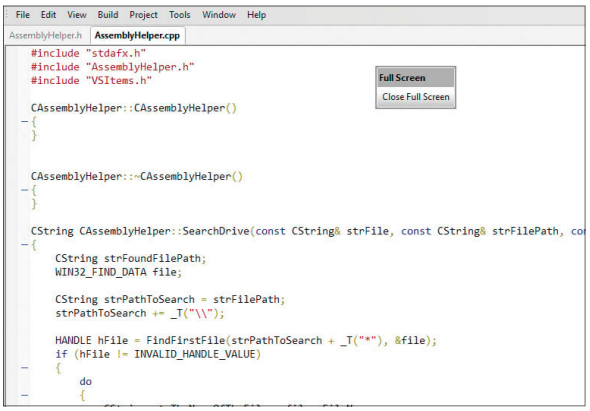
Bref, un développement propre prend du temps. Le fait de développer dans l'IDE requiert des fonctionnalités importantes et plein de petites choses qui font que le logiciel est de bonne qualité.

Conclusion

Pour rappel, cet IDE est une application que j'ai développée pour ma fille Lisa qui vient d'avoir 12 ans et j'ai commencé à lui enseigner les bases du C# : déclarer un int, une string, faire une boucle for, faire une opération mathématique complexe et l'exécuter 100 000 fois et vérifier que cela met moins d'une seconde ; bref les bases du code. Je dois avouer que mon défi est réussi car c'est utilisable. Maintenant, je vais lui faire découvrir les joies du code. Faire un foreach sur une collection, stocker des données dans un fichier. Bref, ma version personnelle « d'apprendre le code ». C'est tellement stimulant qu'elle s'y est prise au jeu. Elle a compris que l'ordinateur PC, cette grosse calculette qui vient avec Windows est pour les créateurs. Et donc, elle va s'éduquer via mon IDE léger et j'en suis fier. L'IDE est sur GitHub donc si vous voulez le customiser, faites-vous plaisir ! •







• Sébastien Putier
Consultant .Net
Ai3 / Agence Est

C# 7.1 : quoi de neuf ?

Après 7 versions majeures du langage C#, chacune apportant des nouveautés très importantes pour le quotidien du développeur, Microsoft a sorti dans le courant du mois d'août 2017 une première version mineure numérotée 7.1.

Note

Cet article fait suite au dossier C# 7.2 & 8.0 publié dans Programmez ! n°213

L'apparition d'une première mise à jour mineure montre une volonté de Microsoft d'accélérer le rythme de publication des nouvelles fonctionnalités pour le langage C#. Ce dernier s'enrichit dans cette nouvelle mouture de trois nouveautés simples. Elles ne révolutionnent pas la vie du développeur, mais apportent de quoi lui simplifier la vie.

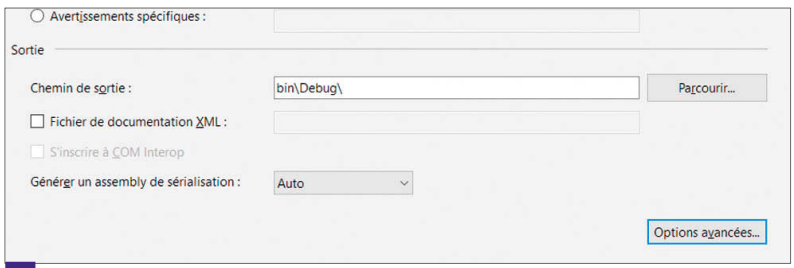
Utiliser le compilateur C# 7.1

Cette version de C# est supportée et intégrée au sein de Visual Studio 2017 à partir de l'Update 15.3. Pour bénéficier des différentes évolutions intégrées au langage, il faut l'indiquer explicitement pour chaque projet.

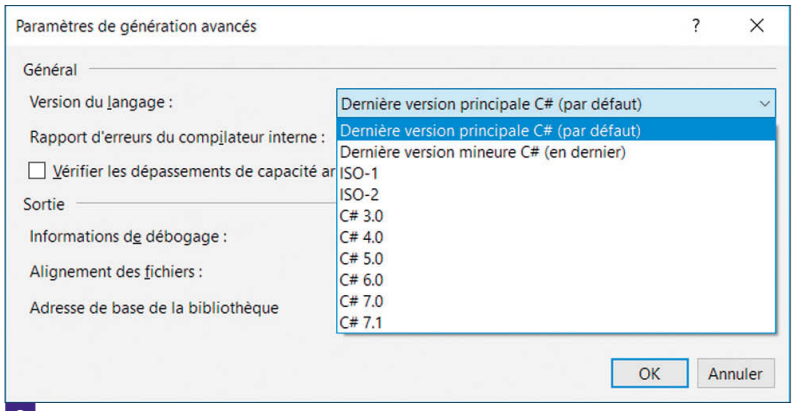
Pour cela, il faut naviguer jusqu'à la section Génération (ou Build) de la fenêtre de propriétés d'un projet C#.

Le bouton Propriétés avancées situé à la fin de la section ouvre une fenêtre modale. 1 et 2

Par défaut, la version du langage sélectionnée est « Dernière version principale C# », c'est-à-dire la dernière version majeure supportée, soit 7.0. La sélection de « Dernière version mineure C# » permet quant à elle d'utiliser la version 7.1, puis 7.2 dès que celle-ci sera disponible sur la machine.



1 *Options avancées*



2 *Choix de version*

Méthode Main asynchrone

Depuis l'intégration de l'asynchronisme au sein même du langage avec les mots-clés async et await, les développeurs manipulant la méthode Main de leurs applications sont accoutumés à l'utilisation du type de code suivant :

```
static void Main(string[] args)
{
  WaitAsync().GetAwaiter().GetResult();
}

private static async Task WaitAsync()
{
  await Task.Delay(1);
}
```

Pourtant, on pourrait se dire que le moyen le plus simple d'exécuter WaitAsync de manière asynchrone est l'utilisation du mot-clé await au cœur de la méthode. En fait, jusqu'à C# 7.0, ceci n'est juste pas possible : la méthode Main ne peut tout simplement pas être déclarée async.

C# 7.1 change la donne à ce niveau en autorisant l'existence d'un point d'entrée asynchrone dans l'application : la méthode Main peut être déclarée avec le mot-clé async 3

L'intégration de cette fonctionnalité est clairement le point majeur de la version. Bon nombre de développeurs découvrant les joies d'async et await se trouvaient en effet plus ou moins bloqués (vive StackOverflow !) dès les premières lignes de code à cause de l'impossibilité d'écrire une méthode Main async… Ils peuvent désormais tirer parti très simplement de l'asynchronisme intégré au langage.

```
class Program
{
    static async void Main(string[] args)
    {
        await WaitAsync();
    }

    private static async Task WaitAsync()
    {
        await Task.Delay(1);
    }
}
```

3 *Async Main*

L'expression default et l'inférence de type

L'utilisation du mot-clé default, déjà présent au sein des blocs switch, a été enrichie dès C# 2 pour offrir la possibilité d'obtenir la valeur par défaut d'une instance d'un type générique d'une manière "passe-partout".

```
public T GetDefaultValue<T>()
 => default(T);
```

Cette syntaxe peut se révéler un peu lourde lors d'une utilisation intensive : il faut obligatoirement passer le type pour lequel on veut récupérer la valeur par défaut. C# 7.1 introduit la notion d'inférence de type pour les expressions default. Lorsque le type associé à l'expression default peut être extrapolé par le compilateur à partir du contexte de son appel, il devient facultatif. La méthode GetDefaultValue ci-dessus peut donc être légèrement simplifiée :

```
public T GetDefaultValue<T>()
 => default;
```

Cette évolution peut également être mise à profit dans le contexte d'une portion de code non-générique.

```
public struct Point
{
   public float X { get; set; }
   public float Y { get; set; }
}

public class MyClass
{
   private Point _point = default;
}
```

L'inférence de nommage sur les tuples

Les tuples sont des ensembles de valeurs potentiellement hétérogènes qui sont associées. Ils sont représentés au sein du framework .NET par le type générique ValueTuple, et sont intégrés à C# 7.0 à l'aide d'une syntaxe qui leur est spécifique : une suite de valeurs (variables ou non) séparées par des virgules, le tout encadré par des parenthèses.

```
var monTuple = ("ma première valeur", 42, DateTime.Now, "ma quatrième valeur") ;
```

La variable résultante a la structure suivante : 4

Comme le montre l'illustration précédente, les différentes valeurs du groupement sont associées à des membres nommés Item1, Item2, Item3 et Item4. Bien que les tuples soient très pratiques pour créer des groupes de données éphémères (comme valeur de retour d'une fonction, par exemple), le nommage qu'ils utilisent par défaut n'est pas forcément idéal lorsque l'on cherche à produire un code lisible.

```
Console.WriteLine($"{monTuple.Item1} – {monTuple.Item2}");
```

Pour cette raison, il est également possible d'associer un nom à chacune des valeurs.

```
var monTuple2 = (Prenom : "Gaël", Age : 3, DateNaissance : new DateTime (2014, 05, 28));
```

5

Les noms des membres étant ainsi plus explicites, l'utilisation des tuples s'en trouve facilitée. On se retrouve toutefois avec une déclaration de variable plus verbeuse, ce qui va à l'encontre de l'idée de départ, à savoir garder un code lisible.

```
Console.WriteLine($"{monTuple2.Prenom} – {monTuple2.Age} ans");
```

C# 7.1 introduit l'inférence de nommage pour les tuples : lorsqu'un tuple est créé à partir de variables existantes, les noms de ses membres sont générés automatiquement à partir du nom des variables sources.

```
string ville = "Lyon";
string codePostal = "69001";
string[] fleuves = new[] { "Rhône", "Saône", "Beaujolais" };
var monTuple3 = (ville, codePostal, fleuves);
```

L'exécution de la portion de code précédente produit une variable avec 3 membres : ville, codePostal et fleuves. 6

Conclusion

L'accélération du cycle des releases du langage C# permet à l'équipe produit d'offrir plus régulièrement de nouvelles fonctionnalités qui, sans être essentielles, rendent plus belle la vie des développeurs. Et Microsoft ne s'arrêtera pas ici : les versions 7.2 et 7.3 sont dans les tuyaux ! •

monTuple	("ma première valeur", 42, {21/11/2017 20:28:00}, "ma quatrième valeur")		(string, int, System.DateTime, string)
Item1	"ma première valeur"		string
Item2	42		int
Item3	{21/11/2017 20:28:00}		System.DateTime
Item4	"ma quatrième valeur"		string

4 *Tuple Items*

monTuple2	("Gaël", 3, {28/05/2014 00:00:00})		(string Prenom, int Age, System.DateTime DateNaissance)
Prenom	"Gaël"		string
Age	3		int
DateNaissance	{28/05/2014 00:00:00}		System.DateTime

5 *Tuple nommé*

monTuple3	("Lyon", "69001", {string[3]})		(string ville, string codePostal, string[] fleuves)
ville	"Lyon"		string
codePostal	"69001"		string
fleuves	{string[3]}		string[]
[0]	"Rhône"		string
[1]	"Saône"		string
[2]	"Beaujolais"		string

6 *Tuple inférence*

Sur abonnement ou en kiosque

Le magazine des pros de l'IT

Mais aussi sur le web

L'INFORMATICIEN
www.linformaticien.com

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

Coder un sine scroll sur Amiga 500

Partie 2

Cet article est le deuxième d'une série de cinq consacrés à la programmation d'un one pixel sine scroll sur Amiga, un effet très utilisé par les coders de démos et autres cracktros sur Amiga… jusqu'à ce qu'il soit passé de mode, étant mis à portée de n'importe quel lamer par le fameux DemoMaker du groupe Red Sector Inc., ou RSI (Figure 1).

Note

Cet article se lit mieux en écoutant l'excellent module composé par Nuke / Anarchy pour la partie magazine de Stolen Data #7, mais c'est affaire de goût personnel…

Dans le premier article, nous avons vu comment installer en environnement de développement sur un Amiga émulé avec WinUAE et coder la Copper list de base pour afficher quelque chose à l'écran.

Ce mois-ci, nous allons voir comment préparer une police de caractères 16x16 pour en afficher facilement les colonnes de pixels des caractères, précalculer les valeurs du sinus requises pour déformer le texte en modifiant l'ordonnée des colonnes, et mettre en place un triple buffering pour alterner proprement les images à l'écran. Vous pouvez télécharger l'archive contenant le code et les données du programme présenté ici à l'URL suivante :

http://www.stashofcode.fr/code/coder-une-one-pixel-sine-scroll-sur-amiga/sinescroll.zip

Cette archive contient plusieurs sources :

- sinescroll.s est la version de base dont il sera question jusqu'à ce que nous optimisions ;
- sinescroll_final.s est la version optimisée de la version de base ;
- sinescroll_star.s est la version enjolivée de la version optimisée.

Générer une police 16x16 aux pixels bien ordonnés

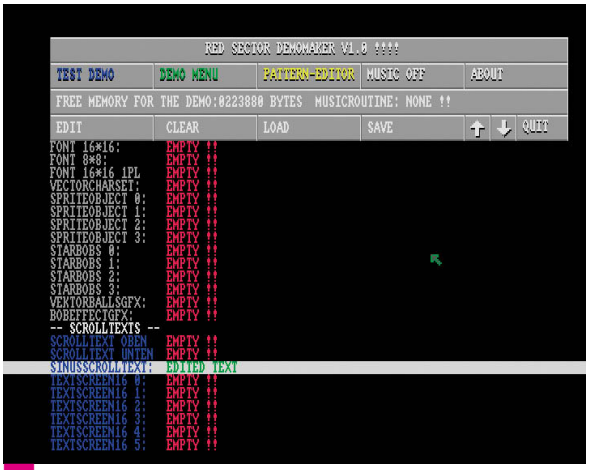
A défaut de grives, on se contente de merles. N'ayant pas retrouvé dans nos archives un fichier correspondant à une police 16x16, nous allons utiliser une police 8x8 dont nous allons doubler les dimensions. Le résultat ne rendra pas justice à la finesse de l'affichage du sine scroll, mais cela permettra de d'avancer.

Le fichier font8.raw contient la police 8x8. Au passage, c'est avec la directive INCBIN qu'on indique à ASM-One qu'il doit lier le code compilé avec des données lues dans un fichier :

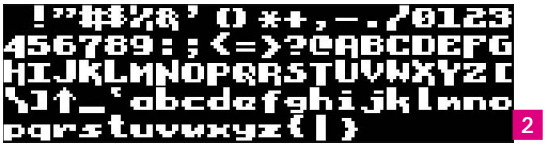
```
font8:    INCBIN "sources:2017/sinescroll/font8.fnt"
```

C'est une suite des 94 caractères ASCII donnés dans l'ordre, chaque caractère se présentant sous la forme d'une matrice de 8x8 bits dont les octets sont donnés dans l'ordre – le bitmap du caractère sur un bitplane. Une représentation permet de se faire une idée de la police, sans donc refléter l'organisation de ses données en mémoire (Figure 2).

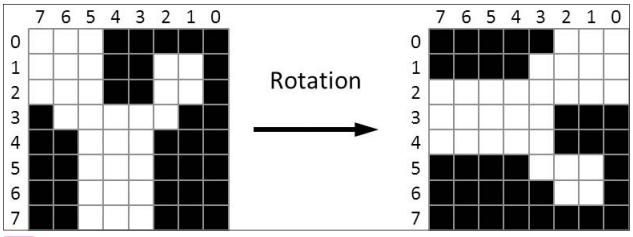
Pour produire un one pixel sine scroll, nous devons dessiner les colonnes de pixels d'un caractère à des hauteurs variables. Pour dessiner la colonne N, nous n'allons certainement pas perdre du temps à lire la ligne 0, en extraire la valeur du bit N, dessiner ou effacer le pixel correspondant à l'écran, et répéter tout cela pour les 7 autres lignes. Nous voulons lire en une fois les 8 valeurs du bit N formant la colonne N. Cela impose d'appliquer une rotation de -90 degrés au bitmap (Figure 3).



1 *Le DemoMaker de Red Sector Inc., ou RSI (cf. lien utile à la fin de cet article).*



2 *Les 94 caractères de la police 8x8, affichés sur plusieurs lignes.*



3 *Transformation de la police 8x8 en vu d'un affichage par colonne.*

Au passage, comme nous souhaitons disposer d'une police 16x16, nous doublons chaque ligne et chaque colonne (Figure 4).

Il en résulte le code suivant, somme toute assez simple :

```
    lea font8,a0
    move.l font16,a1
    move.w #256-1,d0
_fontLoop:
    moveq #7,d1
_fontLineLoop:
    clr.w d5
    clr.w d3
```

```
    clr.w d4
_fontColumnLoop:
    move.b (a0,d5.w),d2
    btst d1,d2
    beq _fontPixelEmpty
    bset d4,d3
    addq.b #1,d4
    bset d4,d3
    addq.b #1,d4
    bra _fontNextPixel
_fontPixelEmpty:
    addq.b #2,d4
_fontNextPixel:
    addq.b #1,d5
    btst #4,d4
    beq _fontColumnLoop
    move.w d3,(a1)+
    move.w d3,(a1)+
    dbf d1,_fontLineLoop
    lea 8(a0),a0
    dbf d0,_fontLoop
```

Dans ce code, la nouvelle police 16x16 est stockée à l'adresse contenue dans `font16`. C'est un espace mémoire de 256*16*2 octets que nous avons alloué plus tôt : 256 caractères de 16 lignes de 16 pixels (2 octets) chacun, dont le contenu se présente finalement comme une succession de caractères retournés (Figure 5).

Alterner proprement les images à l'écran

Après l'initialisation vient la boucle principale. Sa structure est particulièrement simple :

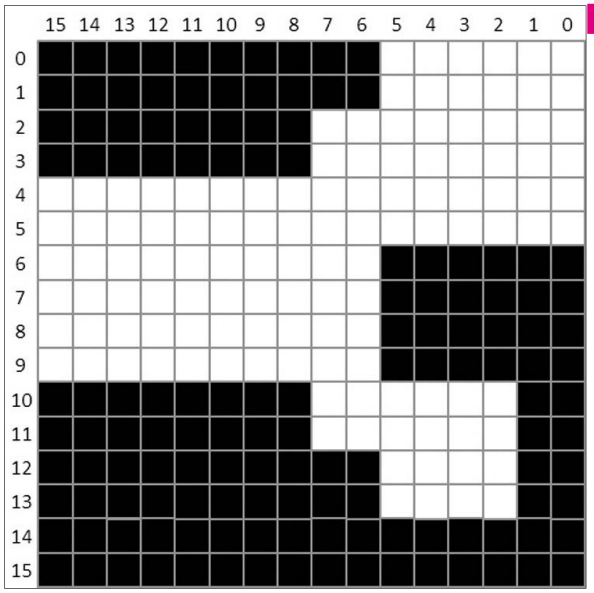
- attendre que le faisceau d'électrons ait atteint (il n'y a pas d'éventualité puisqu'on attend que ce soit fait pour faire la suite…) le bas de la fenêtre d'affichage ;
- dessiner le texte à partir de la position courante et incrémenter cette position ;
- tester si l'utilisateur clique le bouton gauche de la souris, et sinon boucler.

Pour attendre le faisceau d'électrons, il suffit de lire sa position verticale sur 9 bits dans VPOSR pour le bit 8 et VHPOSR pour les bits 0 à 7. Il est important de ne pas se contenter des bits 0 à 7, car affichant en PAL, nous pourrions décider de spécifier une hauteur de l'écran telle que la position `DISPLAY_Y+DISPLAY_DY` dépasse `$FF`. C'est d'ailleurs bien le cas, puisque `$2C+256` donne `300` …

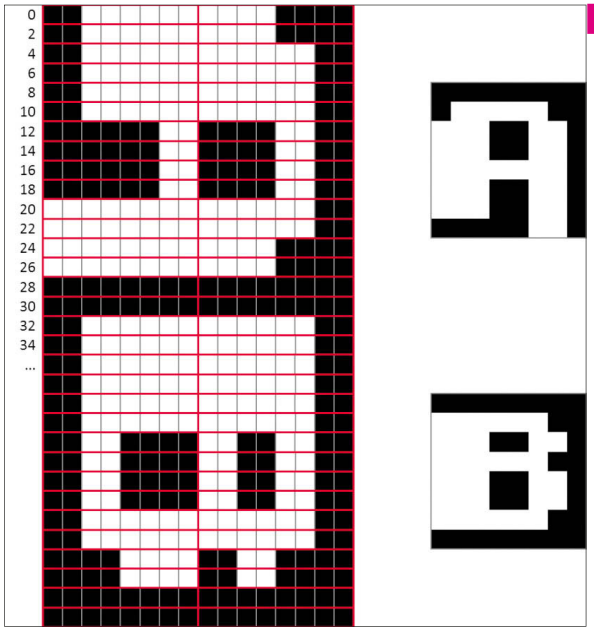
Il serait possible d'attendre que le hardware signale la fin de la trame en testant le bit VERTB qu'il positionne dans INTREQ, puis en effaçant ce bit pour acquitter – le hardware n'efface jamais un bit qu'il a positionné dans INTREQ :

```
_loop:
    move.w INTREQR(a5),d0
    btst #5,d0
    bne _loop
    move.w #$0020,INTREQ(a5)
```

Toutefois, cela ne surviendrait qu'au dépassement de la ligne 312 (la dernière des 313 lignes du PAL), soit bien après la ligne `DISPLAY_Y+DISPLAY_DY` à partir de laquelle nous n'avons plus rien à



4 *Agrandissement des caractères.*



5 *La police 16x16 en mémoire (2 octets par ligne).*

dessiner. Mieux vaut donc attendre que le faisceau d'électrons atteigne cette dernière ligne pour enclencher sur le rendu de la trame suivante. Autant de temps de gagné !

Le code présume que le temps pris par une itération de la boucle principale dépasse celui pris par le faisceau d'électrons pour tracer les lignes `DISPLAY_Y+DISPLAY_DY` à 312. Si ce n'était pas le cas, il faudrait rajouter un test pour attendre le faisceau à la ligne 0. Cela permettrait de ralentir la boucle principale pour rester calé sur la fréquence d'une trame, c'est-à-dire tous les 50èmes de seconde en PAL. Pour tester si l'utilisateur clique le bouton de la souris gauche, il suffit de tester le bit 6 de CIAAPRA, un registre 8 bits d'un des 8520 qui contrôlent les entrées et les sorties dont l'adresse est `$BFE001`.

```
_loop:

_waitVBL:
    move.l VPOSR(a5),d0
    lsr.l #8,d0
```

```
and.w #$01FF,d0
cmp.w #DISPLAY_Y+DISPLAY_DY,d0
blt _waitVBL

;Code à exécuter dans la trame ici

btst #6,$bfe001
bne _loop
```

Ce cadre étant posé, les choses sérieuses peuvent commencer. La première tâche à accomplir en début de trame consiste à afficher le bitplane dans lequel nous avons dessiné le sine scroll durant la trame précédente. C'est le principe du double buffering : ne jamais dessiner dans le bitplane affiché pour éviter le flicker.
Toutefois, nous n'allons pas nous contenter de faire du double buffering. En effet, comme le Blitter dispose d'un DMA, il est capable d'effacer un bitplane tandis que le CPU dessine dans un autre et que le hardware en affiche un troisième. C'est du triple buffering (Figure 7). Après permutation circulaire des trois bitplanes… :

```
move.l bitplaneA,d0
move.l bitplaneB,d1
move.l bitplaneC,d2
move.l d1,bitplaneA
move.l d2,bitplaneB
move.l d0,bitplaneC
```

…il suffit donc de modifier l'adresse du bitplane à afficher là où elle utilisée pour alimenter BPL1PTH et BPL1PTL dans la Copper list :

```
movea.l copperlist,a0
move.w d1,9*4+2(a0)
move.w d1,10*4+2(a0)
swap d1
move.w d1,11*4+2(a0)
move.w d1,12*4+2(a0)
```

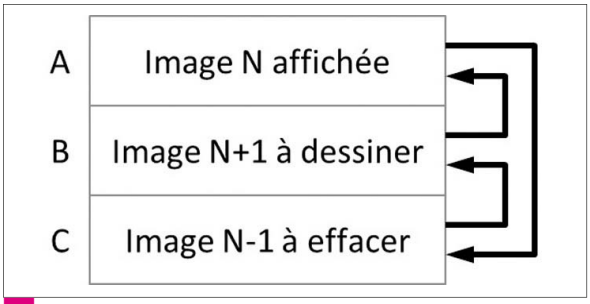
Dès lors, il est possible de lancer l'effacement du bitplane précédemment affiché au Blitter :

```
WAITBLIT
move.w #0,BLTDMOD(a5)
move.w #$0000,BLTCON1(a5)
move.w #$0100,BLTCON0(a5)
move.l bitplaneC,BLTDPTH(a5)
move.w #(DISPLAY_DX>>4)!(256<<6),BLTSIZE(a5)
```

Comme cela a été expliqué dans un article précédent, le Blitter peut combiner logiquement bit à bit des blocs de mémoire sources dans un bloc de mémoire destination. Pour cela, il se base sur une formule qu'il faut décrire en positionnant des bits dans BLTCON0, une combinaison logique par OR de combinaisons logiques par AND des données provenant des sources A, B et C, éventuellement inversées par NOT – par exemple, `D=aBc+aBC+AB+ABC`, ce qui revient à `D=B`, c'est-à-dire à copier
dans le bloc de mémoire destination D le bloc mémoire source B. Si nous spécifions `D=0` par omission de tous les autres termes qui peuvent composer la formule, nous demandons donc au Blitter de remplir de 0 le bloc de mémoire destination.
Le Blitter fonctionnant en parallèle du CPU, nul besoin d'attendre qu'il termine d'effacer l'image courante de l'animation du sine scroll dans son bitplane pour commencer à dessiner l'image suivante dans un autre bitplane. Il sera toujours temps d'attendre le Blitter quand nous voudrons nous en servir pour dessiner un caractère en lui demandant… de tracer des lignes !

Déformer selon un sinus

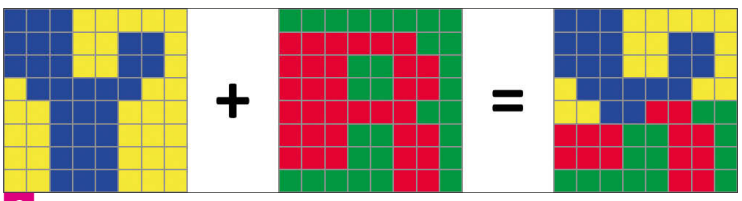
Le principe du one pixel sine scroll consiste à dessiner les 16 colonnes de pixels qui composent un caractère à des hauteurs différentes, ces dernières étant calculées en fonction du sinus d'un angle incrémenté entre deux colonnes.
La formule pour calculer l'ordonnée de la colonne x est donc:
$y = $ `SCROLL_Y` $+ ($ `SCROLL_AMPLITUDE` $>>1)*(1+\sin(\beta_x))$
…où β_x est la valeur de l'angle β pour la colonne x, incrémenté de `SINE_SPEED_PIXEL` à la colonne suivante.
Noter que l'amplitude des ordonnées est alors [-`SCROLL_AMPLITUDE>>1` ,] `SCROLL_AMPLITUDE>>1`, ce qui correspond à une hauteur de `SCROLL_AMPLITUDE` +1 si `SCROLL_AMPLITUDE` est paire et de `SCROLL_AMPLITUDE` si cette valeur est impaire.



7 *Permutation circulaire des bitplanes en triple buffering.*

FLICKER ET DOUBLE BUFFERING

Supposons qu'il s'agisse de commencer à dessiner un « R » rouge sur fond vert par-dessus un « Y » bleu sur fond jaune au moment où le faisceau d'électrons commence à afficher les bitplanes. Il est possible que le CPU se mette à dessiner le « R » à un moment où le faisceau a déjà affiché une bonne partie du « Y », et allant plus vite que le faisceau d'électrons, qu'il modifie une partie des bitplanes que ce dernier n'a pas encore affichés. En conséquence, le faisceau commence à lire les données des bitplanes après et non avant que le CPU les ait modifiés, et donc à afficher une partie du « R » dans la foulée d'une partie du « Y » (Figure 6).



6 *Flicker lors du dessin dans le bitplane affiché.*

Quand l'image affichée change à chaque trame, cette course entre le faisceau d'électrons et le CPU produit un chevauchement d'images successives à partir d'une position qui généralement varie : c'est le flicker.
Pour l'éviter, il faut dessiner le « R » dans des bitplanes cachés, différents de ceux où le « Y » est dessiné, pour leur part affichés. Quand la trame se termine, il faut demander l'affichage des bitplanes où le « R » a été dessiné et commencer à dessiner la lettre suivante dans les bitplanes où le « Y » est dessiné. C'est le double buffering.

Par exemple, avec SCROLL_Y=0, SCROLL_AMPLITUDE=17 et SINE_SPEED_PIXEL=10 (Figure 8).

Oops ! nous avons oublié un détail : la fonction sin() ne figure pas dans le jeu des instructions du 68000. Comme il est hors de question d'appeler une fonction d'une librairie se livrant à des calculs coûteux en cycles d'horloge du CPU, nous allons précalculer les valeurs du sinus pour tous les angles de 0 à 359 degrés par pas de 1 degré. Autrement dit, nous allons les tenir à disposition sous la forme d'une table prête à l'emploi.

Re-oops ! nous avons oublié un autre détail : nous ne savons pas gérer les nombres à virgule flottante. Pour les mêmes raisons, nous allons précalculer les valeurs du sinus sous la forme d'entiers. Et comme l'amplitude des valeurs est [-1, 1], nous allons devoir multiplier ces valeurs par un facteur; sans ce facteur, elles se limiteraient à -1, 0 et 1. Bref, pour reprendre des formules Excel, nous allons calculer ARRONDI(K*SIN(A);0), K étant le facteur et A étant l'angle.

Ce facteur, nous n'allons pas le choisir par hasard. En effet, puisqu'une valeur du sinus sera utilisée dans une multiplication, l'opération devra être suivie d'une division par le facteur en question. Toujours dans un souci d'économie, nous excluons de recourir à l'instruction DIVS, très coûteuse en cycles. Ce sera un décalage arithmétique de bits sur la droite, c'est-à-dire une division entière signée par une puissance de 2. Le facteur doit donc prendre la forme 2^N.

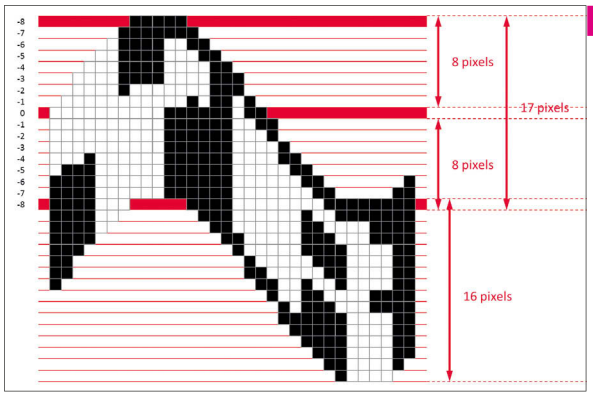
Nous choisissons de fixer N à 15 pour une excellente précision des valeurs du sinus et la possibilité d'utiliser une instruction SWAP (décalage de 16 bits sur la droite, donc division par 2^16) suivie d'une instruction ROL.L d'un bit (décalage d'un bit sur la gauche, donc multiplication par 2), ce qui sera plus économique en cycles qu'un ASR.L de 15 bits. La table du sinus se présente alors ainsi :

```
sinus:  DC.W 0          ;sin(0)*2^15
        DC.W 572     ;sin(1)*2^15
        DC.W 1144       ;sin(2)*2^15
        DC.W 1715       ;sin(3)*2^15
        ;...
```

Il reste un petit problème à régler. En effet, effectuer une multiplication signée par 2^N entraîne un débordement de 16 bits quand la valeur du sinus est -1 ou 1. Ainsi, 1*32768 donne 32768, valeur signée 17 bits qui ne peut donc tenir dans la table des valeurs signées 16 bits du sinus. Dans ces conditions, cette table ne pourrait pas contenir des valeurs correspondant exactement aux conversions de -1 et 1, mais seulement très proches approximations : -32767 et 32767.

Nous décidons de ne pas tolérer cette imprécision, et c'est pourquoi N est réduit de 15 à 14, quand bien même cela nous contraint à faire suivre le SWAP d'un ROL.L de 2 bits au lieu d'un. Démonstration que cela tient aux limites :

```
move.w #$7FFF,d0      ;2^15-1=32767
move.w #$C000,d1      ;sin(-90)*2^14
muls d1,d0            ;$E0004000
swap d0               ;$4000E000
rol.l #2,d0       ;$00038001 => $8001=-32767 OK

move.w #$7FFF,d0      ;2^15-1=32767
```



8 *Déformation verticale des caractères selon un sinus.*

```
move.w #$4000,d1      ;sin(90)*2^14
muls d1,d0            ;$1FFFC000
swap d0               ;$C0001FFF
rol.l #2,d0       ;$00007FFF => $7FFF=32767 OK
```

In fine, la table du sinus se présente ainsi :

```
sinus:  DC.W 0          ;sin(0)*2^14
        DC.W 286 ;sin(1)*2^14
        DC.W 572 ;sin(2)*2^14
        DC.W 857 ;sin(3)*2^14
        ;...
```

Le résultat 16 bits de la multiplication d'une valeur signée 16 bits stockée dans D1 par le sinus d'un angle exprimé en degrés stocké dans D0 s'obtient ainsi :

```
lea sinus,a0
lsl.w #1,d0
move.w (a0,d0.w),d2
muls d2,d1
swap d1
rol.l #2,d1
```

Définissons SCROLL_DY comme la hauteur de la bande que le sine scroll peut occuper à l'écran. Partant, SCROLL_AMPLITUDE doit être telle que (SCROLL_AMPLITUDE >>1)*(1+sin($_x$)) génère des valeurs comprises dans [0, SCROLL_DY -16]. Cela n'est possible que si cet intervalle comprend un nombre impair de valeurs, donc si SCROLL_DY -16 est paire. Cela tombe bien, car nous souhaitons que le scroll soit centré verticalement à l'ordonnée SCROLL_Y, ce qui impose que SCROLL_DY soit paire, car DISPLAY_DY, la hauteur de l'écran, est paire. Ce qui donne :

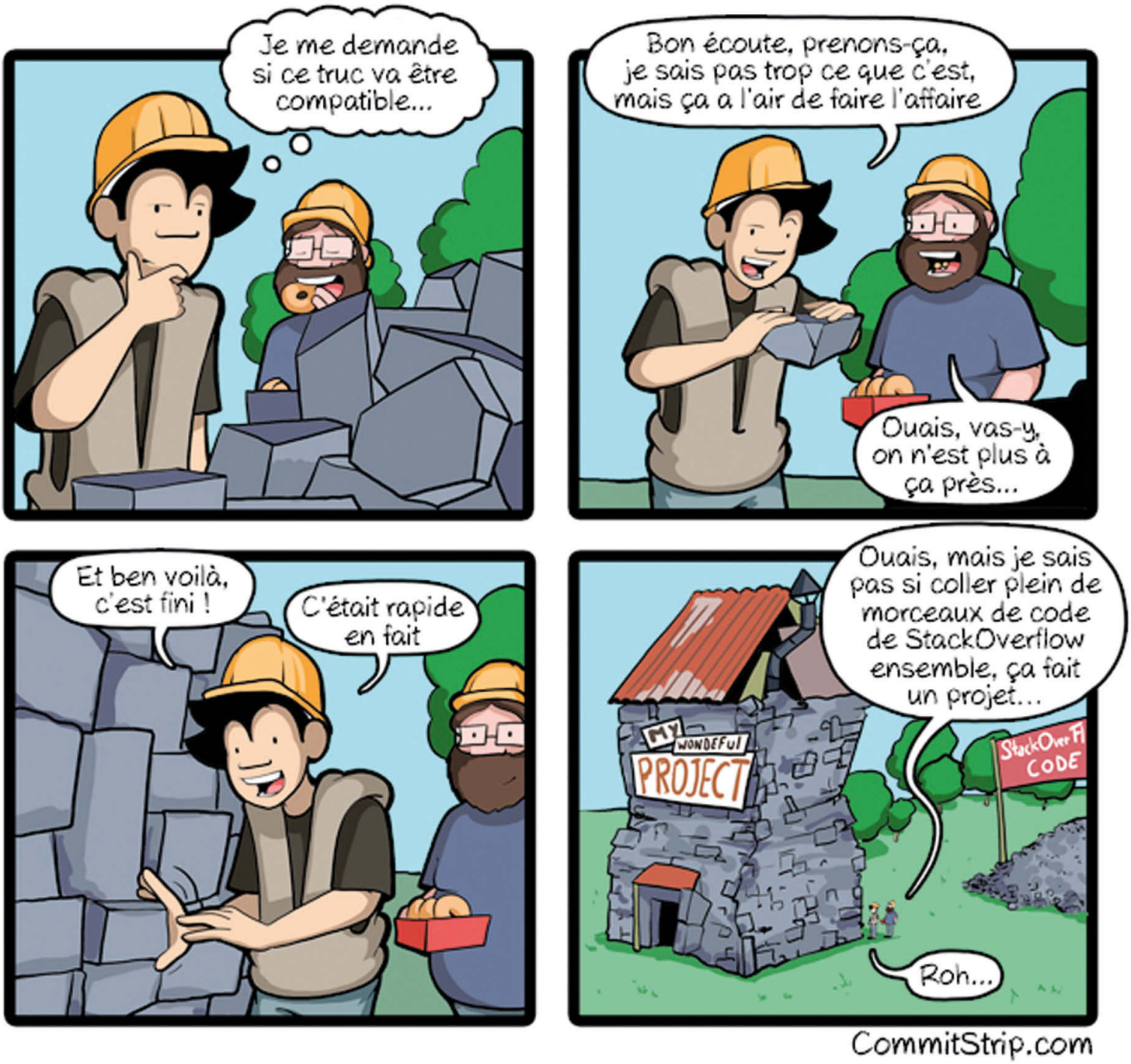
```
SCROLL_DY=100
SCROLL_AMPLITUDE=SCROLL_DY-16
SCROLL_Y=(DISPLAY_DY-SCROLL_DY)>>1
```

Tant que nous y sommes, nous définissons des constantes définissant l'abscisse à laquelle scroll démarre et le nombre de pixels sur lequel il s'étend. Par défaut, ce sera toute la largeur de l'écran :

```
SCROLL_DX=DISPLAY_DX
SCROLL_X=(DISPLAY_DX-SCROLL_DX)>>1
```

L'ordonnée de chaque colonne d'un caractère du sine scroll pouvant être calculée, il est désormais possible de dessiner ce dernier. Auparavant, il faut mettre en place son défilement et son animation… •

stack overflow patchwork



Une publication Nefer-IT, 57 rue de Gisors, 95300 Pontoise - redaction@programmez.com
Tél. : 09 86 73 61 08 - Directeur de la publication & Rédacteur en chef : François Tonic
Secrétaire de rédaction : Olivier Pavie
Ont collaboré à ce numéro : S. Saurel
Nos experts techniques : G. Duval, E. Lenoir, B. Renault, S. Keller, F. Santin, X. Blanc, J. Paquet, A. Vache, S. Labourey, P. Charrière, G. Cottrez, F. Pigere, D. Van Robaeys, O. Philippot, S. Putier, P. Giry, C. Martraire, C. Pichaud, D. Duplan
Couverture : © Orla - **Maquette** : Pierre Sandré.
Publicité : PC Presse, Tél.: 01 74 70 16 30, Fax : 01 40 90 70 81 - pub@programmez.com.
Imprimeur : S.A. Corelio Nevada Printing, 30 allée de la recherche, 1070 Bruxelles, Belgique.
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 -
Publicité : pub@programmez.com - Evenements / agenda : redaction@programmez.com

Dépôt légal : à parution - Commission paritaire : 1220K78366 - ISSN : 1627-0908 - © NEFER-IT / Programmez, janvier 2018

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

CACD2

La manufacture digitale
du groupe Crédit Agricole

NOUS RECRUTONS !

Développeurs
FRONT-END
BACK-END
MOBILE

Nous recrutons actuellement des profils sur les technos Java, .Net, Node.js, Kotlin, Swift, Javascript, Angular, Php, React, HTML5/CSS3…

LA MANUFACTURE DIGITALE

Face à l'évolution digitale de ses métiers, le Groupe Crédit Agricole transforme une ancienne imprimerie en une structure entièrement dédiée à la production digitale: [CACD2].

Située dans un ancien lieu industriel à Paris 14ème, la structure met le "faire" au coeur de son fonctionnement.

POUR PLUS D'INFORMATIONS, CONTACTEZ-NOUS :

recrutement@cacd2.fr

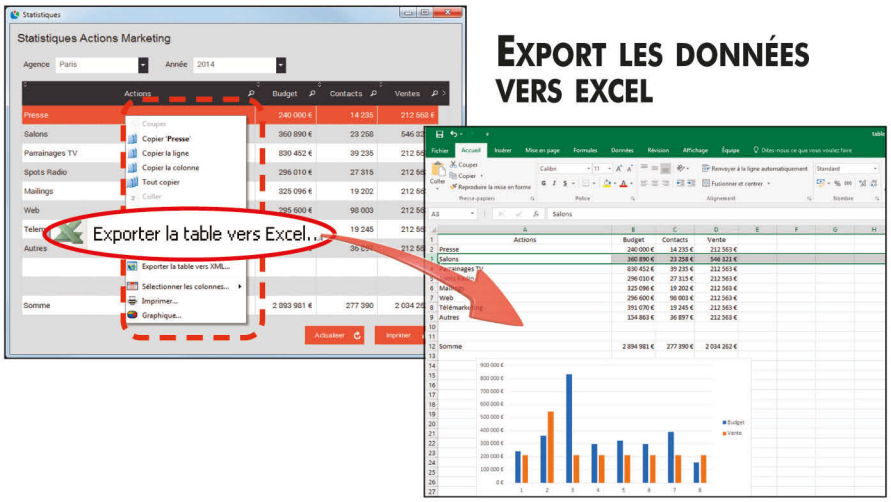
www.cacd2.fr

CACD2

VOUS UTILISEZ DES APPLICATIONS DÉVELOPPÉES EN WINDEV. CONNAISSEZ-VOUS CES FACILITÉS ?

Grâce à la technologie des FAA («Fonctionnalités Automatiques des Applications» incluse en standard dans toutes les applications développées avec WINDEV, vous bénéficiez de dizaines de fonctionnalités utiles au quotidien. En voici 4 présentées ci-dessous. A tout endroit d'une fenêtre, utilisez le clic droit de votre souris pour découvrir des fonctionnalités insoupçonnées et gagner beaucoup de temps.

EXPORT LES DONNÉES VERS EXCEL



EXPORT AUTOMATIQUE VERS WORD, EXCEL ET OPEN OFFICE

Dans chaque application WINDEV un menu contextuel est automatiquement présent (accessible via le clic droit de la souris) et permet un export vers Excel, Word, PDF, XML... depuis les tables visualisées.

Pour l'équipe de développement: aucune programmation n'est nécessaire, il s'agit d'une fonctionnalité standard. Si nécessaire, un mot de passe peut être demandé. Vos applications développées avec WINDEV sont automatiquement riches de fonctionnalités utiles. Notez que chaque FAA peut être débranchée par programmation.

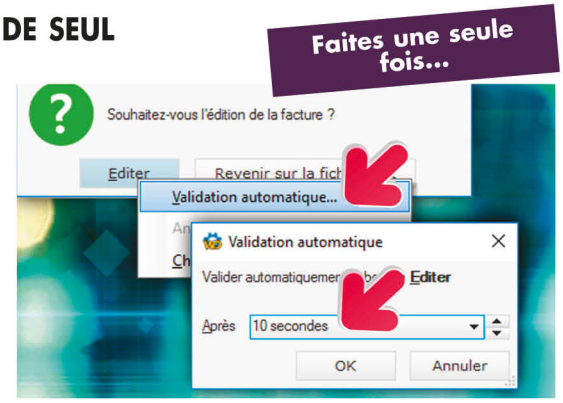
LE BOUTON SE VALIDE SEUL

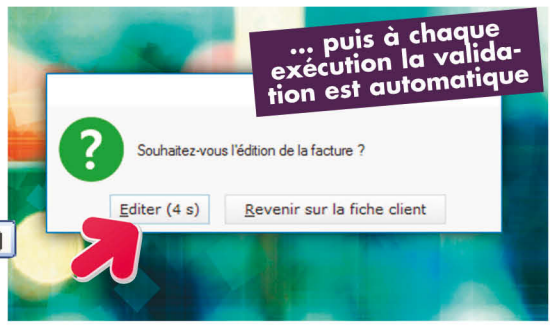
BOUTON MINUTERIE: UN TRAITEMENT NE SERA JAMAIS BLOQUÉ !

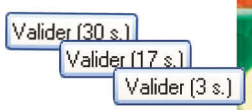
Le bouton Minuterie affiche un décompte en secondes et se valide automatiquement à la fin de ce décompte, sauf si vous l'interrompez.

C'est très utile pour ne pas bloquer un traitement par lots (batch) qui pose des questions de temps en temps !

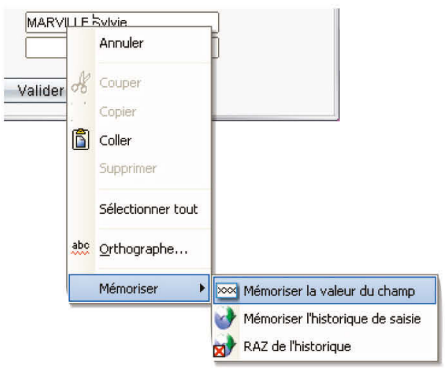
Ajouter cette minuterie sur n'importe quel bouton est facile grâce à WINDEV: un simple clic droit suffit pour indiquer le délai de validation automatique.







PRÉ-REMPLIR UN CHAMP



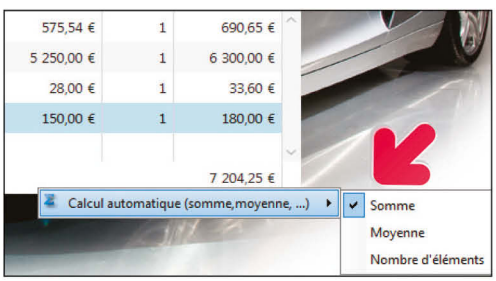
PRÉ REMPLISSAGE DES CHAMPS

Pour ne pas avoir à re saisir une valeur habituelle, vous pouvez activer la persistance des champs: mémoriser une valeur fixe, ou rappeler la dernière valeur saisie.

Très utile pour les fourchettes de dates par exemple ou pour saisir une valeur identique dans un même champ, ce qui arrive souvent !

En combinant persistance de champs et bouton à validation automatique, il est ainsi possible de lancer automatiquement des traitements récurrents, comme de statistiques ou des éditions.

AJOUTEZ UN TOTAL



AJOUT D'UN TOTAL, D'UNE MOYENNE...

Il arrive qu'un tableau soit visualisé sans total de colonne dans une application.

Grâce aux FAA de WINDEV, pas de souci !

D'un simple clic-droit, rajoutez un total (ou une moyenne, ou un compteur...) là où vous le désirez.

WINDEV®

www.pcsoft.fr