programmez.com
[Programmez!]
Le magazine des développeurs
215 FÉVRIER 2018

Intelligence artificielle

Le développeur au cœur du futur

© Petmal

Le top 10 des failles de sécurité

WebExtensions
Un standard pour étendre les navigateurs

Retour terrain : Pydio
Migrer le code PHP en Go : un défi technique

Android
Développez un clone de Flappy Bird

M 04319 - 215S - 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

SERVEURS DÉDIÉS

XEON®

AVEC ikoula HÉBERGEUR CLOUD

Optez pour un serveur dédié dernière génération et bénéficiez d'un support technique expérimenté.

debian ubuntu CentOS Windows Server 2012

POUR LES LECTEURS DE **PROGRAMMEZ***

OFFRE SPÉCIALE -60 %

À PARTIR DE

11,99€ HT/MOIS

~~29,99€~~

CODE PROMO **XEPRO17**

- Assistance technique **en 24/7**
- Interface **Extranet** pour gérer vos prestations
- **KVM sur IP** pour garder l'accès
- Analyse et surveillance **de vos serveurs**
- **RAID Materiel** en option
- Large choix d'**OS** Linux et Windows

*Offre spéciale -60 % valable sur la première période de souscription avec un engagement de 1 ou 3 mois. Offre valable jusqu'au 31 décembre 2017 23h59 pour une seule personne physique ou morale, et non cumulable avec d'autres remises. Prix TTC 14,39 €. Par défaut les prix TTC affichés incluent la TVA française en vigueur.

CHOISSISEZ VOTRE XEON®

https://express.ikoula.com/promoxeon-pro

ikoula HÉBERGEUR CLOUD /ikoula @ikoula sales@ikoula.com 01 84 01 02 50

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

EDITO

La dictature du framework

Le framework, voilà un composant technique qui s'est totalement imposé aux développeurs. Difficile aujourd'hui de ne pas utiliser un framework quand on développe sur desktop, cloud, web, mobile et même mainframe. Il est là pour vous simplifier la vie, du moins, vous faciliter des aspects de la programmation, éviter de tout réécrire, alléger le code et les implémentations. Il offre une couche d'abstraction bienvenue.

"Tous les six mois, un nouveau (framework) apparaît, affirmant qu'il révolutionne le développement de l'interface utilisateur. Des milliers de dévs l'adoptent dans les nouveaux projets, et des posts de blogs sont écrits, les questions sur Stack Overflow apparaissent, puis un nouveau (encore plus révolutionnaire) framework surgit et usurpe le trône" (Ian Allen, blog Stack overflow).

Avant de crier au buffer overflow, ce n'est pas totalement faux, même si une poignée de frameworks dominent le marché… Mais cela ne signifie pas que dans 5 ans ils seront encore là; Zend Framework était incontournable dans le monde PHP. Symfony est arrivé, et la suite vous la connaissez.

Parfois, le framework impose des changements radicaux. L'histoire d'Angular illustre bien la cassure entre les premières versions JavaScript et le passage à TypeScript.De nombreux débats ont agité la communauté.

La question se pose sur le suivisme à faire sur les évolutions constantes des frameworks : faut-il toujours être à jour ? Mais vous pouvez aussi avoir des frameworks qui évoluent tous les 2-3 ans. On a le temps de consolider ses développements, au risque d'attendre trop longtemps des ajouts de nouvelles fonctions ou des améliorations attendues. Par contre quand le framework utilisé n'est plus maintenu, bah là, pas le choix, on migre vers un autre framework et on croise les doigts pour que tout ne fonctionne pas trop mal…

Et finalement, quand on n'a qu'un seul framework possible, on n'est pas content. Et quand on peut choisir parmi 10, on n'est pas content non plus. ☺

François Tonic
ftonic@programmez.com

SOMMAIRE

Dans le prochain numéro !
Programmez! #216, dès le 2 mars 2018

CHOISIR SA BASE DE DONNÉES SQL, NOSQL
VISUAL STUDIO CODE :
l'IDE open source de Microsoft
LES NOUVEAUTÉS D'ANGULAR 5

Zones blanches :

promis, juré, les zones blanches dans le réseau mobile seront un mauvais souvenir d'ici 3-4 ans. Les opérateurs ont accepté d'investir 3 milliards et d'installer 5 000 nouvelles antennes (chaque opérateur)… Et 10 000 communes, d'ici 2020, seront couvertes par la 4G. Chaque année, le gouvernement promet la fin des zones blanches. L'espoir fait vivre.

Cortana va-t-elle suivre le destin de Windows Mobile ? C'est la conclusion de certains observateurs après le CES de janvier. Non, il ne faut pas enterrer Cortana. Mais Microsoft devra convaincre bien mieux qu'actuellement face au rouleau compresseur Alexa d'Amazon. Google cherche activement à challenger Amazon.

GoPro connaît encore des difficultés : 20 % des effectifs à la porte, arrêt des drones ! Et le chiffre d'affaire se prend une claque : -37 % sur 1 an.

Kaby-Lake G est une gamme un peu étrange. Elle est le résultat du travail commun entre Intel et AMD pour intégrer un GPU Vega sur un processeur Core i7 et i5. Cette étonnante fusion est une bonne nouvelle car les puces graphiques Intel n'étaient pas toujours très performantes. Ces puces seront intégrées dans les futurs barebones Intel NUC 8. Ces modèles succèdent aux excellents Skull.

HTML 5.2 arrive en version définitive

HTML 5.2 voit sa spécification définitive publiée sur le site du W3C. Une des nouveautés est l'API Payment Request qui doit permettre de réduire les fraudes et les erreurs de paiement en ligne. Autre nouveauté, l'apparition de l'ARIA. Il s'agit de recommandations pour l'accessibilité aux apps web. Le système de plugins est désormais obsolète.

Apple casse le tarif du changement de batterie : 29 € . Les accusations d'obsolescence programmée et de ralentissement volontaire de certains modèles (même si au départ, la proposition technique est intéressante) ont fait réagir la Pomme.

Le film sur Han Solo

sera-t-il un succès ou une catastrophe ? Depuis plusieurs semaines, des sources Disney évoquent de gros soucis sur le film, voire un échec prévisible. Attendons déjà la sortie du film…

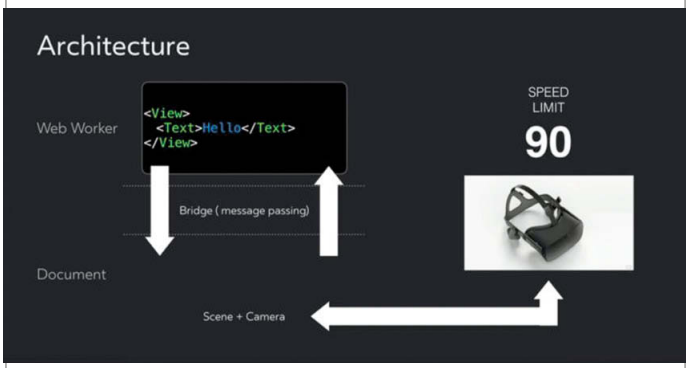
Argh ! Notre bon roi Arthur a terminé le scénario de Kaamelott mais le budget n'est pas encore bouclé.

Star Trek sauce Tarantino : voici une idée excitante et inquiétante mais le résultat peut être explosif !

ARKit : un premier bilan

La réalité augmentée sur smartphone n'a jamais été un argument de vente ni un support excitant pour cette technologie. Malgré les belles promesses d'ARKit et ARCore, il semble qu'Apple ait encore du mal à convaincre. Les apps ARKit sont finalement peu nombreuses à en croire Apptopia, à peine 1 000 apps. Les débuts furent plutôt bons mais les nouveautés se sont rapidement essoufflées. La technologie doit trouver son rythme de croisière et Apple doit continuer à en parler et à montrer, en attendant un matériel dédié. Sans surprise, les jeux pèsent 30 %, l'éducation est finalement peu présente, avec moins de 8 %.

2018 : l'année de REACT ?



Quelle sera la grande tendance JavaScript et des frameworks pour cette année ? Eric Elliot a fait une petite compilation des frameworks phares pour cette nouvelle année : https://medium.com/javascript-scene/top-javascript-libraries-tech-to-learn-in-2018-c38028e028e6

REACT devrait gagner encore plus en popularité, confirmant sa popularité en 2017 et les bons retours.

Vue.JS, mine de rien, gagne aussi en popularité même si ce n'est pas le plus voyant des frameworks JS. Bonne visibilité sur Github : téléchargement et notation en hausse. Une des critiques est son apprentissage mais là, chacun se fera une opinion. Une tendance à suivre en 2018.

REACT reçoit visiblement un excellent taux de satisfaction, par rapport à Vue.JS, qui est déjà excellent. L'un des avantages de REACT est la grande diversité : React Native, React pour IoT, React pour la réalité virtuelle. Angular reste une valeur sûre même s'il s'éloigne de JavaScript. A suivre aussi cette année.

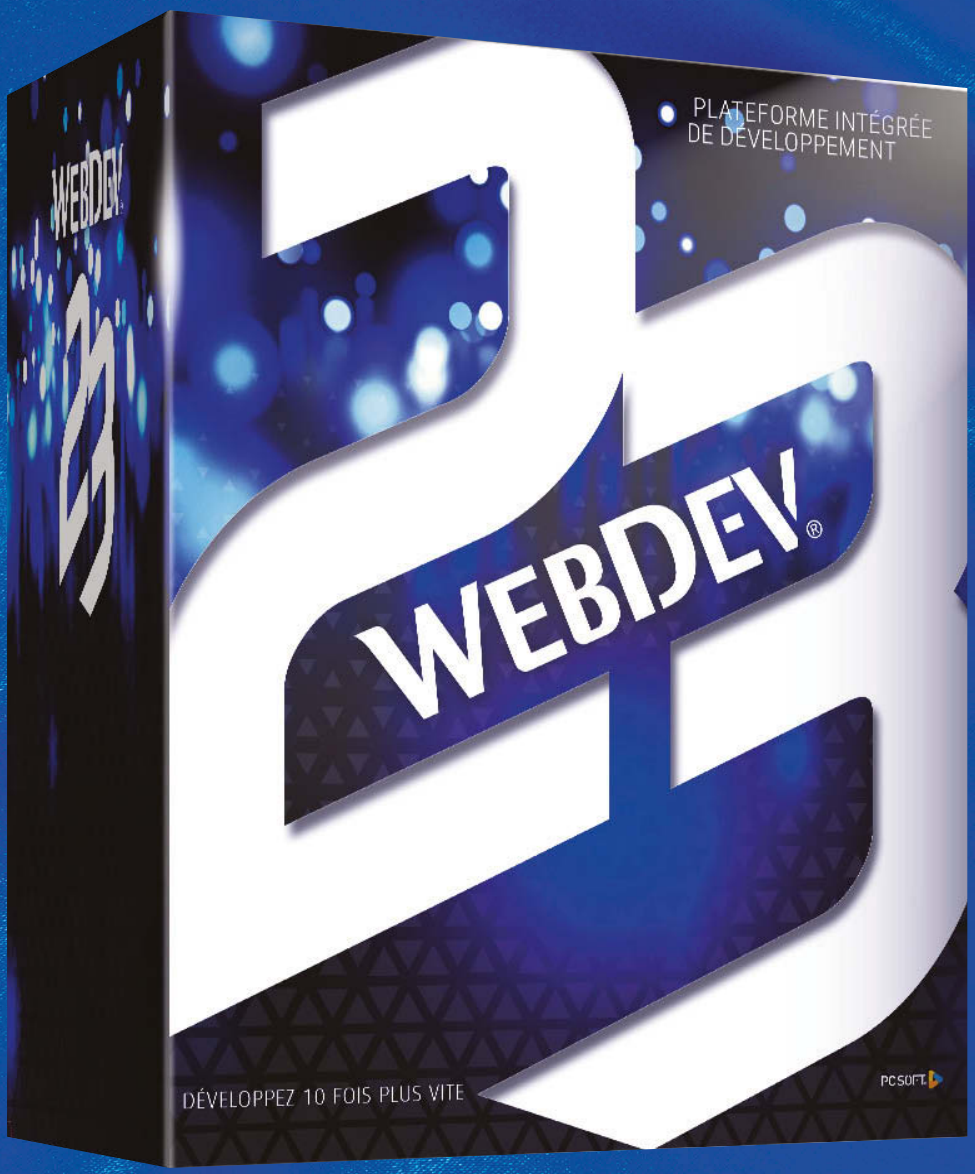
Ces frameworks auront aussi un impact sur les compétences recherchées par les entreprises. Vue.JS sera forcément moins demandé car encore trop peu connu en France contrairement à Angular et React.

Le top 10 des langages selon Spectrum (2017)

Il n'y a pas que l'index TIOBE, très perfectible, pour avoir un classement même approximatif des langages les plus utilisés. IEEE Spectrum croise une dizaine de sources pour établir un classement. On citera les tendances dans les recherches ou encore les offres d'emplois.

Classement	Langage	Domaines
1	C	mobile, entreprise, embarqué
2	Java	web, mobile, entreprise
3	Python	web, entreprise
4	C++	mobile, entreprise, embarqué
5	R	entreprise
6	C#	web, mobile, entreprise
7	PHP	web
8	JavaScript	web, mobile
9	Ruby	web, entreprise
10	Go	web, entreprise

Swift arrive 14e, Scala 16e, VB, 18e, Cobol 41e


WEBDEV
PLATEFORME INTÉGRÉE
DE DÉVELOPPEMENT
WEBDEV®
DÉVELOPPEZ 10 FOIS PLUS VITE
PCSOFT

COMMUNAUTÉS

Annoncez vos meetups, conférences sur Programmez ! : ftonic@programmez.com

Ch'ti JUG

8 février : Continuous Delivery.

GDG Toulouse

8 février : Redux / NGRX & MOBX.

GDG Code d'Armor

20 février : l'orchestration pour les nuls.

GDG Nantes

15 février : Elexis.

React.js and React Native

Meetup le 10 février (Paris).

ParisJUG

13 février : gestion de la mémoire.
13 mars : JEE.
15 mai : spécial 10 ans de ParisJUG !

MARS

Microsoft Tech Summit : 14 & 15 mars – Paris

Participez à des workshops gratuits, animés par la crème des ingénieurs à l'origine de nos solutions Cloud sur Azure et Microsoft 365. Grâce au Microsoft Tech Summit, enrichissez vos compétences, approfondissez votre expertise et allez plus loin dans votre carrière professionnelle.

LLVM, Clang, lldb, lld, Polly : 27 mars - Paris

Le prochain meetup de la communauté se tiendra fin mars. Le thème n'est pas encore connu.

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 Start-up 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

Oracle Cloud 2018

Oracle aime les développeurs et veut le prouver avec cette journée 100 % code, 100 % développeur. L'évènement se déroulera dans plusieurs villes dans le monde dont Paris. Nous y trouvons des sessions, des ateliers. La date française n'est pas encore connue.

JUIN

DevFest Lille

Le Devfest Lille 2018 se passera le 21 juin 2018 sur une seule journée pour 400 personnes prévues. La conférence prendra place dans les locaux de IMT Lille-Douai (20 Rue Guglielmo Marconi, 59650 Villeneuve-d'Ascq). Le site est accessible via https://devfest.gdglille.org/.

Le CFP (http://devfestlille.cfp.io/) est en cours jusqu'au 1er avril 2018 (pour un programme disponible début mai).

Hack in Paris

Du 25 au 29 juin, Hack in Paris sera la grande conférence sur la sécurité et le hacking en France.

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

Mostafa Amokhtari
Directeur technique
France CA Technologies

Dans la famille DevOps, où est passé le testeur ?!

Dans la continuité d'un article publié en juin 2016 dans Programmez intitulé « Dans la famille DevOps, je voudrais le développeur ! », nous souhaitons cette fois-ci aborder le sujet des tests dans une stratégie DevOps avec l'évolution du rôle des testeurs mais également des développeurs dans la recherche de la qualité et de la sécurité des applications.

Dans un marché extrêmement changeant et face à une compétition accrue, la promesse du DevOps associée aux méthodologies Agile et Lean est de pouvoir délivrer des applications logicielles de qualité à une cadence élevée tout en maîtrisant les coûts. Mais quel est l'impact d'une stratégie DevOps sur les équipes de tests et le rôle du testeur ? Quelles sont les nouvelles tendances en matière de tests logiciels ? Comment se positionnent les testeurs dans un environnement Agile ? Quelle est cette nouvelle génération de testeurs dont nous avons besoin ? Nous tenterons d'amener un début de réponse dans cet article sachant que comme le marché, les stratégies évoluent rapidement.

L'activité des tests logiciels connaît depuis quelques années des bouleversements majeurs au niveau des pratiques et des méthodologies de tests. Les facteurs qui induisent ces changements sont, vous l'avez deviné, les mêmes, peu ou prou que ceux à l'origine de la mise en œuvre de stratégie Agile et DevOps dans les entreprises :

- une compétition accrue où le client/utilisateur est non seulement « roi » mais très au fait des usages de l'informatique dans la vie quotidienne ;
- des demandes de changements de plus en plus fréquentes, avec un besoin de réalisation rapide et nécessitant une collaboration plus étroite et constante avec le métier ;
- une recherche de « payback » ou retour sur investissement à très court terme.
- une utilisation exponentielle d'applications mobiles qui deviennent rapidement la vitrine de l'entreprise et dont le développement se fait sur des cycles extrêmement courts...

Dans ce contexte, l'approche traditionnelle du test, quel que soit le nom donné à l'équipe (Contrôle Qualité, Homologation, Cellule de Test ...) a vécu. Les tests planifiés en fin de cycle et réalisés par une équipe dédiée, ne sont plus compatibles avec le planning, la fréquence de livraison souhaitée, et la qualité perçue du service ou produit livré. De plus, en raison des organisations, le testeur est souvent perçu comme un juge qui contrôle et valide le travail du développeur, ce qui ne facilite pas forcément la collaboration.

Plusieurs pistes sont disponibles pour une nouvelle approche des tests que nous regroupons ici sous le terme « Test Agile ».

Les tests comme base de spécification.

Traditionnellement, les métiers émettent des exigences ou spécifications. Ensuite les développeurs et les testeurs, chacun de leur côté, essaient de comprendre ces spécifications. Ceci pour pouvoir produire, dans un cas le code correspondant, et dans l'autre, les cas de test pour pouvoir le tester. Outre un travail en partie redondant et chronophage, plus de 60% des anomalies, selon une étude (Hyderabad Business School GITAM University, « Quality Flaws: Issues and Challenges in Software Development », 2012), sont liées à une mauvaise compréhension des exigences.

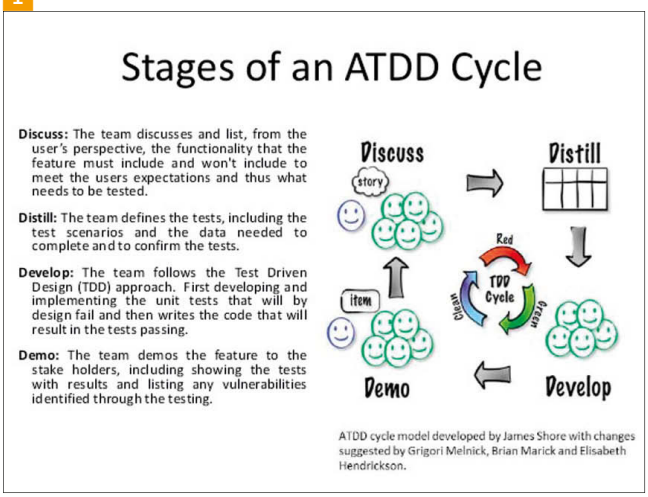
Une approche Agile s'appuyant sur une des quatre valeurs du Manifesto, à savoir « favoriser les individus et leurs interactions plus que les processus et les outils », est de faire cette analyse du besoin de façon conjointe (Métier, Testeur, Développeur). Cette collaboration permet de s'assurer une compréhension commune des spécifications, et de lever les ambiguïtés sur les termes utilisés. En adoptant le formalisme adéquat, les exigences sont transposées immédiatement sous forme d'un ou plusieurs tests en langage naturel.

C'est le principe des méthodologies ATDD (Acceptance Test Driven Developpement), BDD (Behavior Driven Development) et Model-Based Testing toutes basées sur un principe fondamental : les tests sont au cœur du projet. En effet, les tests servent aux spécifications générales du logiciel : ils sont donc créés et automatisés en amont du développement, et orienteront ce dernier. Les tests sont également exécutés automatiquement et permettent de définir lorsque le code est terminé. 1

https://www.slideshare.net/StephenTucker4/agile-and-atdd-the-perfect-couple

Il est important que le testeur s'approprie la syntaxe propre à la démarche ATDD outillée. En effet par ce biais il est possible d'automatiser directement les tests. En complément, le MBT (Model Based

1



NE RATEZ AUCUN NUMÉRO

Abonnez-vous !

[Programmez!] Le magazine des développeurs

Nos classiques

1 an ..... 49€*
11 numéros

2 ans ..... 79€*
22 numéros

Etudiant ..... 39€*
1 an - 11 numéros

** Tarifs France métropolitaine*

Abonnement numérique

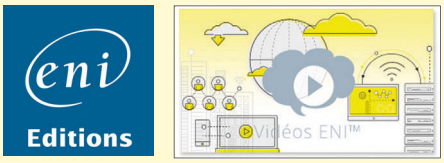
PDF ..... 35€
1 an - 11 numéros
Souscription uniquement sur
www.programmez.com

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



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 215
Offre limitée, valable jusqu'au 2 mars 2018

☐ Mme ☐ M. Entreprise : Fonction :

Prénom : Nom :

Adresse :

Code postal : Ville :

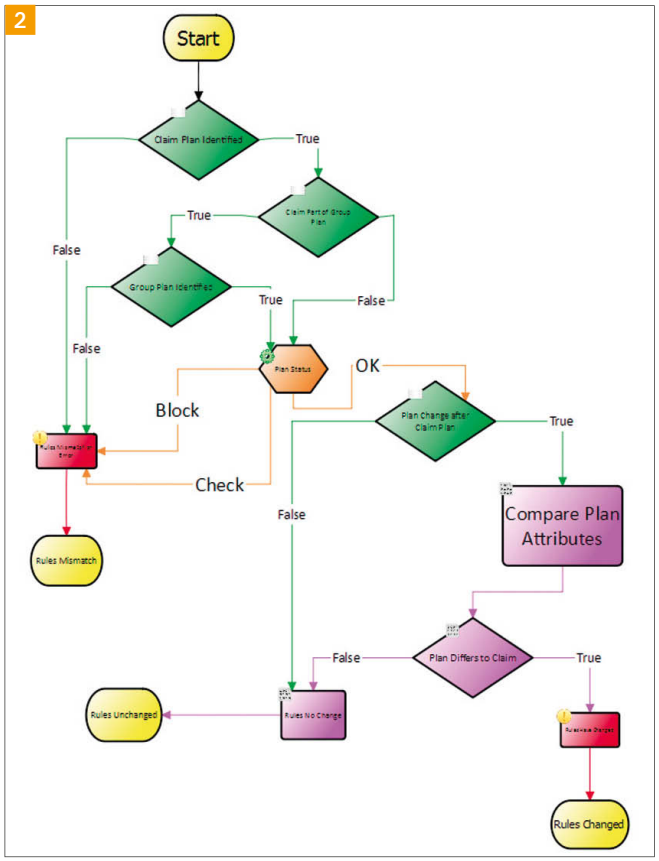
email indispensable pour l'envoi d'informations relatives à votre abonnement

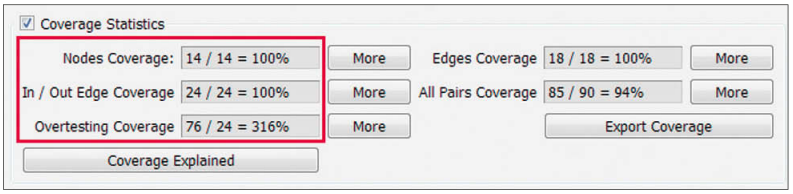
E-mail : @

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

*** Tarifs France métropolitaine**

2





Capture de chemin de tests fonctionnels et niveau de couverture

Testing), en plus de générer les cas et plans de tests, évalue le niveau de couverture fonctionnel et optimise le nombre de tests à lancer en fonction des modifications fonctionnelles réalisées. 2

Les approches TDD pour les tests unitaires et BDD pour les tests fonctionnels sont ainsi complémentaires.

Un avantage non négligeable d'effectuer un développement piloté par les tests est la réduction du volume de code généré au strict minimum, pour que les tests soient passant. Par conséquent, pas de code « mort » ou inutile qui peut s'avérer être une source de problèmes dans le futur.

Les tests en continu.

Dans le cadre de développement en mode Agile avec des itérations rapides, il est fondamental de tester en continu pour s'assurer que le nouveau code généré n'introduise pas d'anomalies. Plus une anomalie est découverte tard dans le cycle de développement, plus elle coûte cher. (Pour un coût de 1 en dev, la même anomalie découverte en production peut coûter entre 100 et 1000 fois plus cher). Le test en continu est finalement un socle indispensable à la mise en place d'une stratégie DevOps.

L'automatisation des tests et le déploiement automatique des systèmes sous test (SUT) sont clés dans ces conditions, mais ne seront possibles qu'accompagnés d'autres éléments :

Des développements et des tests sans contraintes : pour réaliser les tests, le système sous test est souvent dépendant d'un écosystème pas toujours disponible. La virtualisation de services permet de s'affranchir des contraintes d'indisponibilités et de simuler simplement et rapidement le fonctionnel, les données et les performances de tout flux au-dessus de TCP. Elle permet de simuler, à moindre coût, la « vraie vie » telle que vécue en production par une version précédente de l'application, ou bien de développer de nouveaux modèles à partir d'une feuille blanche pour un tout nouveau système.

Des données disponibles : pouvoir accéder à la demande à des données de test de qualité est une condition sine qua non pour réaliser des tests en continu probants. Le défi est de posséder les données pertinentes et spécifiques pour réaliser des tests rigoureux en parallèle et à la demande, tout en respectant les règlementations sur les données personnelles. Des solutions de Gestion de la donnée au niveau entreprise existent comme CA TDM pour localiser ou créer automatiquement des données de test en fonction des tests devant être exécutés.

L'intégration de tests non-fonctionnels aux tests continus

Le principe de base des tests en continu est d'effectuer, le plus en amont possible, des tests à chaque fois qu'un changement est apporté à l'application. Cependant, si le dispositif de Test en Continu n'inclut pas les tests non-fonctionnels, comme les tests de performances et de sécurité, l'équipe ne résout qu'une partie de l'équation. Tous les problèmes de cette nature découverts tardivement peuvent réduire à néant les gains de productivité obtenus grâce au test en continu d'un point de vue fonctionnel, et même compromettre le calendrier des versions.

Grâce aux solutions de test en mode SaaS, il est possible de réaliser des tests de performances très tôt dans le cycle de développement sans pour autant avoir les contraintes de mise à dispositions des ressources correspondantes. CA Blazemeter permet de faire des tests de charge en se basant sur les scripts écrits dans JMeter ou Selenium par exemple.

De même, la qualité du code sur les aspects de sécurité applicative peut être testée par les développeurs en amont, et directement à partir de leur IDE avec des solutions comme GreenLight de Veracode qui proposent également des tests de pénétrations en mode SaaS.

Il est à noter que certains aspects du test resteront manuels comme les tests exploratoires ou d'IHM (ergonomie, « look and feel », …).

Plus que jamais, le rôle du testeur est primordial dans la mise en œuvre d'une stratégie DevOps. Cependant son rôle évolue : plus proche des métiers, son impact business sera prépondérant. Avec une connaissance du métier et des qualités rédactionnelles de test pertinent en ADTT outillé, il sera un maillon incontournable de la chaîne DevOps.

Avec l'automatisation des tests, il devra également avoir une fibre de développeur pour créer et maintenir des tests pertinents. La même rigueur dans le « versionning » du code doit être appliquée aux tests qui évolueront en même temps que l'application.

Plus collaboratif, il sera parfaitement intégré au sein des équipes agiles par ces compétences en « T », c'est-à-dire expert d'un domaine mais compétent dans plusieurs autres, et sera perçu à juste titre comme un « enabler » de qualité.

•

François Tonic

Pydio passe à Go et aux microservices

Pydio est une plateforme de partage de fichiers, ressemblant à Dropbox, Box ou ownCloud. Ce projet est totalement open source, en licence AGPL et existe depuis plus de 10 ans. Le projet connaît une refonte majeure du code grâce au langage Go.

Pour faire simple, Pydio est une infrastructure à déployer pour gérer le partage de fichiers, la synchronisation des fichiers, essentiellement dédiée aux entreprises. Depuis plusieurs mois, les développeurs de la solution redéveloppent le cœur même de la solution. « A l'origine, notre stack technique était du LAMP standard. Ce qui a assuré notre succès. Mais depuis presque 18 mois, nous explorons le langage Go. PHP est au cœur de notre solution, il gère les bases et les données structurées. Par contre, si on déploie avec des milliers d'utilisateurs, nous rencontrons des difficultés. Le système subit alors de fortes charges. Et PHP n'est pas toujours adapté pour gérer des fichiers. Il y a un an nous avons regardé de près Go pour fournir une app compagnon tout en laissant l'enveloppe PHP. » précise Charles du Jeu (CEO / CTO Pydio). Le succès de ce premier contact avec Go a conduit courant 2017 à la décision de migrer entièrement en langage Go le backend, en s'appuyant sur une architecture microservice.

Le microservice comme unité indépendante

L'approche microservice permet de rendre indépendant chaque service les uns des autres. Chaque service communique via des API, ce qui apporte une grande flexibilité en environnement distribué. "On déploie autant de noeuds que nécessaire pour monter en charge. On peut faire évoluer chaque service de manière indépendante" précise Charles. La base technique repose sur le framework micro, écrit en Go. Un des objectifs est d'assurer la découverte automatique des services. Il fournit un bus d'évènements (le bus est la colonne vertébrale indispensable aux microservices). Différents drivers sont implémentables comme consul, nats.io, grpc. Pydio veut du standard, du standard et du standard !

L'usage d'API stables et standard est par contre indispensable. Par exemple, pour la gestion du stockage, il utilise les API d'Amazon S3 ou encore OpenID Connect (basé sur Oauth2) pour assurer l'authentification. Les services communiquent en GRPC, sur http/2. Mais comme on s'en doute, il faut que le backend puisse communiquer avec le frontend, pour cela, les équipes implémentent une API externe. Et finalement, chaque service reçoit un service servant de gateway pour exposer une API en REST : bref, on sépare tout, on expose via une gateway et on redistribue en interne sur les différents services !

Un frontend inchangé faute de temps

"Pour le moment, notre frontend reste en PHP. Nous manquons de temps pour migrer vers Go. Ce sera sans doute pour la prochaine version. PHP représente encore 30 à 40 % de notre code total. Notre interface / ergonomie est beaucoup appréciée de nos utilisateur. Pour ce faire, nous utilisons un client lourd PHP + JavaScript, avec une architecture modulaire par plug-in. Il s'agit donc d'un développement important si nous voulons passer cette partie au langage Go. D'autre part, quand on déploie Pydio, il faut au préalable configurer correctement les services (virtual host, Apache, etc.). La mouture Go doit donc être parfaitement identique et complète" ajoute Charles du Jeu.

Une architecture microservice

"Ce découpage est finalement assez naturel, le code PHP est bien structuré et organisé, avec des notions de services, de gestion des utilisateurs. Donc c'est une migration assez naturelle pour nos développeurs. Il a tout de même fallu revoir toute la gestion des données. Car une des difficultés est de gérer et de supporter les systèmes de fichiers tels Samba, S3, l'objet, etc. Il faut pouvoir accéder aux répertoires locaux. C'est un travail long car il faut (re)définir les spécifications, repenser l'indexation, séparer les sources de données (point d'accès aux données) de la gestion des droits d'accès" commente Charles du Jeu. Le microservice nous permet d'agréger dynamiquement le contenu des data sources sur un gros arbre d'indexation. Par exemple quand les user A et B partagent, on a des flags pour dire au système que l'on partage. "Heureusement, on arrive à se soustraire du système de fichiers. Amazon S3 est un stockage objet par défaut, on pose la couche objet sur le système de fichier local, tout est transparent. " pointe Charles du Jeu.

Cependant, par le passé, l'API S3 a pu poser des soucis car elle évolue et il faut garder le niveau de version pour éviter tout problème. Au lieu de s'appuyer directement sur l'API Amazon, Pydio utilise une API compatible S3; le projet Minio. Les équipes réalisent une veille constante. Mais si Pydio utilise S3, il garde le support d'autres protocoles, comme WebDAV (ce qui n'est pas de gaieté de coeur).

PHP -> Go -> microservices

Passer d'un langage à un autre n'est jamais facile surtout pour un projet en production. Un des points forts de Go, selon Pydio, est le framework de tests unitaires. Et même si le code PHP est considéré comme propre, il a fallu repartir de 0. "Nous avons mis en place une couverture de codes, de l'intégration continue, mais nous n'avons pas encore beaucoup de tests d'intégration. Et nous regardons beaucoup Docker et les conteneurs. Nous voulons éviter la complexité. La philosophie DevOps est un objectif. Sur les conteneurs, nous voulons nous intégrer à Kubernetes pour le déploiement des microservices. Il ne faut pas réinventer la roue. Il existe des outils d'orchestration qui font très bien le travail. Une des difficultés du projet est que nos utilisateurs ont des architectures hétérogènes. Nous souhaitons définir 2-3 architectures de référence et fournir les éléments de déploiement et de production nécessaires. " conclut Charles du Jeu. •

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

François Tonic

Les processeurs sous pression : Meltdown & Spectre font des dégats

Le mois de janvier a été secoué par une affaire aussi sérieuse qu'aux effets profonds : des failles dans les processeurs permettent des attaques et des vols de données. Intel a été mis en cause mais l'affaire s'est rapidement étendue à AMD et ARM et aux éditeurs de systèmes d'exploitation. Les déclarations, les patchs en tous genres se sont multipliés, au risque de créer le chaos. Les experts Google sont à l'origine de ces découvertes l'an dernier.

Meltdown

Spectre

Meltdown et Spectre touchent le cœur de nombreux processeurs. Le problème n'est pas seulement logique mais aussi matériel suite à une faille dans la puce. Meltdown touche les processeurs Intel à architecture x86. Spectre concerne Intel, AMD et ARM (Cortex A). Grosso modo, ces failles font la même chose : permettre l'accès à une partie de la mémoire enfouie dans le processeur et qui normalement constitue une enclave sécurisée et inaccessible. Un des problèmes est l'isolation entre les applications des utilisateurs et le système d'exploitation, car une attaque pourrait accéder à cette mémoire protégée via l'application, permettant ainsi de récupérer les données sensibles stockées dedans. Spectre agit fondamentalement de la même manière mais entre différentes applications.

Meltdown était connu depuis plusieurs mois et ne constitue pas forcément une surprise pour les fondeurs et éditeurs. Spectre est une faille plus complexe à corriger même si un exploit est plus difficile à réaliser. Une correction logicielle est possible mais ne pourra pas garantir une rustine totale à la faille. Le problème est trop profond. La solution ultime sera de changer le processeur par un processeur sain.

Référence de l'annonce par Google : https://googleprojectzero.blogspot.fr/2018/01/reading-privileged-memory-with-side.html

Meltdown : CVE-2017-5754

Comme dit plus haut, cette faille est spécifique aux CPU Intel. Il permet de procéder, illégalement, à une élévation des privilèges. Cela signifie que l'on peut accéder à des mémoires protégées au niveau du noyau. Potentiellement, la faille peut permettre d'attaquer les données sensibles stockées sur la machine proprement dite. Et elle permet aussi d'exécuter du code malveillant pour récupérer, par exemple, l'adressage mémoire allouée au noyau et avoir accès aux données… Le problème devient problématique quand une machine est partagée / mutualisée entre plusieurs utilisateurs comme avec des instances virtualisées… Une des corrections au niveau OS est de mettre en place une isolation Kernel Page Table Isolation pour isoler strictement les espaces mémoires du kernel et ceux des utilisateurs.

Spectre : CVE-2017-5753 & CVE-2017-5715

Cette faille ne se limite pas aux processeurs Intel. Les mécanismes de Spectre sont ceux de Meltdown mais avec une variante : un programme peut accéder aux espaces mémoires d'un autre programme. Cela signifie : accéder aux données de ces espaces. Par exemple, il serait possible de récupérer les données présentes en mémoire vive à partir d'un site web (servant de programme d'attaque). Les correctifs doivent être faits sur la partie logicielle (OS et navigateurs par exemple) mais aussi en corrigeant les processeurs eux-mêmes.

Ces failles exploitent des faiblesses dans les processeurs. Elles s'appuient sur le principe de l'exécution dans le désordre (Out of order execution). Ce modèle permet aux processeurs d'exécuter des instructions le plus efficacement possible et pas forcement dans l'ordre d'arrivée de ces instructions. Or, ce mécanisme précis est en cause. Spectre s'appuie sur une autre méthode d'exécution processeur : l'exécution spéculative. Il s'agit de mécanismes pour améliorer les performances du CPU.

Les définitions données par Check Point :

- **L'exécution spéculative** consiste en le lancement anticipé d'une instruction, afin de pouvoir suivre le rythme des tâches toujours plus nombreuses à exécuter, mais sans s'assurer que cette anticipation soit correcte : soit elle l'est et dans ce cas le résultat est attendu et déjà prêt à être fourni, soit elle n'est pas correcte et il suffit d'ignorer ce résultat sans que cela n'impacte le temps passé ;

- **Les limites de sécurité :** les ordinateurs modernes s'appuient sur leur capacité à supporter des codes distincts simultanément, tout en maintenant des limites. Par exemple, si vous regardez une vidéo YouTube tout en étant connecté sur votre compte bancaire dans un autre onglet, le processeur peut maintenir la séparation entre ces deux espaces. Cependant, l'exécution spéculative se fait parfois sans vérifier les limites de sécurité, toujours dans l'optique de ne pas ralentir le processeur, dans la mesure où les vérifications de sécurité seront réalisées avant que le résultat final ne soit fourni.

Intel est pointé du doigt à cause du nombre de processeurs installés dans les ordinateurs et les serveurs dans le monde entier. Mais il ne faut pas oublier les autres.

Des failles potentiellement dangereuses pour les infrastructures et les systèmes

Ces failles concernent les infrastructures et les serveurs. Dès début janvier, Google a mis en ligne des précisions sur les risques et les procédures déployées par les équipes. Ainsi, les images de Compute Engine ont été patchées pour protéger les utilisateurs. Mais Google précise que les images plus anciennes ne contiennent pas les patchs et que le risque existe.

Chaque éditeur de système doit alors proposer et déployer les patchs de sécurité adéquats. Des mises à jour des firmwares sont déployées.

Les correctifs ont un impact sur les performances, même si l'ampleur des ralentissements n'est pas aussi catastrophique qu'annoncée. Red Hat évoque des impacts de 2 à 19 % selon les mécanismes et les fonctions concernés.

Dans des infrastructures virtualisées, cela signifie qu'il faut patcher les systèmes hôtes et redémarrer chaque hôte. Et faire de même pour chaque guest. Des fournisseurs cloud ont procédé à des dépréciations de systèmes et de noyaux. Par exemple, Google, dans Compute Engine, a déprécié les noyaux 2.6.x et utiliser les noyaux patchés.

Google a été très actif sur ces failles et a publié de nombreux posts sur le sujet. Un long post a été mis en ligne sur les actions faites sur les services cloud et les plateformes Google : https://security.googleblog.com/2018/01/more-details-about-mitigations-for-cpu_4.html

Cette page est très intéressante, car elle évoque les actions faites par Google et celles que les utilisateurs doivent faire. Ainsi, Android a été patché pour réduire les accès. Par contre, pour les anciennes versions, à voir. De même, il faudra bien mettre à jour les navigateurs web, certains objets connectés… Pour ces derniers, ce sera au cas par cas.

Microsoft : des patchs ont été proposés mais des problèmes de redémarrages ou d'impossibilité de démarrer les machines ont forcé l'éditeur à les retirer. Cela concernait les machines utilisant des puces AMD.

Linux : des patchs ont été déployés dans le noyau

Apple : des mises à jour iOS et macOS ont été rapidement proposées mais une certaine confusion régnait sur les failles corrigées.

Quelques gammes de puces Intel concernées : Core i3, i5, i7 ; Xeon séries 3400, 3600, 5500, E3, E5, E7 ; Aton séries C, E, A, x3, Z ; Celeron séries J, N

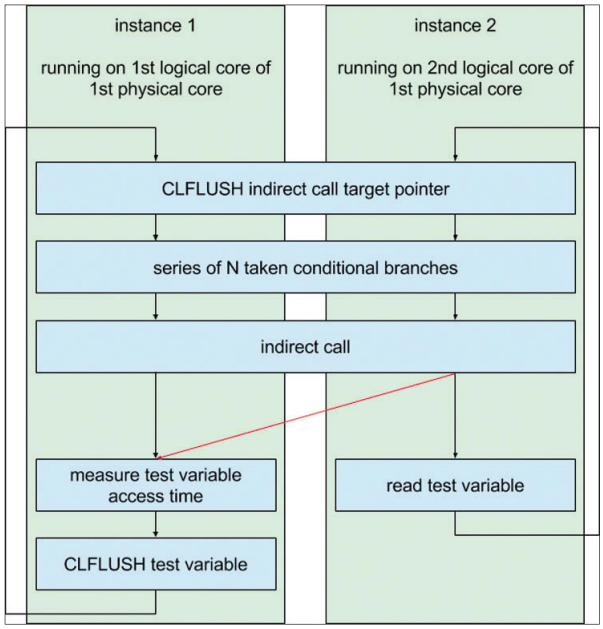
Intel a annoncé des puces corrigées dans les prochaines semaines.

Quelques gammes de puces AMD concernées :Ryzen, FX, Athlon.

Côté ARM, plusieurs architectures sont concernées par les failles : Cortex R8, A8, A9, A15, A17, A57, A72, A75. Mais la vulnérabilité diffère d'une famille à une autre. Mais il semble que les processeurs spécifiques ne sont pas impactés, car les mécanismes en cause ne sont pas toujours présents. Apple a précisé que ses matériels sont aussi concernés, car les processeurs Intel et les dérivés ARM sont utilisés. Qualcomm, silencieux au début, a confirmé qu'il était lui aussi concerné.

Le problème ne pourra être définitivement résolu qu'avec des designs processeurs et des processeurs corrigés. On peut espérer que les futures machines ne sont pas sensibles à ces failles.

Ces failles montrent bien les faiblesses de notre monde technologique. Et surtout, quand ces problèmes sont enfouis dans les mécanismes des processeurs, les corrections sont difficiles, d'autant plus pour corriger le matériel. Ces deux failles ne sont finalement qu'un exemple parmi d'autres que la sécurité totale en informatique n'existe pas et n'existera jamais. Mais imaginons des failles similaires dans des millions d'objets connectés ou au cœur de services cloud…



La fondation Raspberry Pi s'est empressée de dire que la carte n'était pas concernée par ces failles qu'elle appelle simplement bugs. La Pi utilise un design ARM non impacté, voir le post très complet mis en ligne : https://www.raspberrypi.org/blog/why-raspberry-pi-isnt-vulnerable-to-spectre-or-meltdown/

OVH a rapidement appliqué des patchs sur les systèmes utilisés et les services. Ces mises à jour concerne les trois failles CVE : https://docs.ovh.com/fr/dedicated/information-about-meltdown-spectre-vulnerability-fixes/

Ikoula procède aussi à une mise à jour des infrastructures. Pour les serveurs et machines virtuelles dédiés, à l'utilisateur de faire les mises à jour.

N'oublions pas les compilateurs

Les compilateurs modernes optimisent considérablement les codes, en s'appuyant sur les fonctions les plus avancées des processeurs. Ce qui ne va pas sans risque avec ces failles. Pour les compilateurs GCC et LLVM, des correctifs sont disponibles ou le seront très rapidement.

Les équipes de Google ont présenté une technique pour prévenir les injections (faille Spectre) : Retpoline. Référence complète : https://support.google.com/faqs/answer/7625886

Précision de la liste LLVM et l'approche Retpoline : https://lists.llvm.org/pipermail/llvm-commits/Week-of-Mon-20180101/513630.html •

Attention :

Les compilateurs ne vont colmater les failles. Ils réduisent (un peu) les risques, notamment sur Spectre mais cela ne suffira pas même si on modifie les optimisations et l'injection de certains mécanismes processeurs durant la compilation.

Loic Daval
Expert IOT SQLI Lyon

Blockchain et Industrie 4.0, un combo pour l'avenir ?

Vous le savez certainement, les évolutions technologiques permettent de traiter de plus en plus de données. L'essor du big data démultiplie les capacités d'analyse statistique à grande échelle et ouvre de nouvelles voies vers l'intelligence artificielle. La Smart Industrie a tout intérêt à s'emparer au plus vite de cette opportunité pour transformer son activité dont Les enjeux sont multiples : amélioration de la productivité, diminution des temps d'arrêts de production, meilleure sécurité pour les salariés…

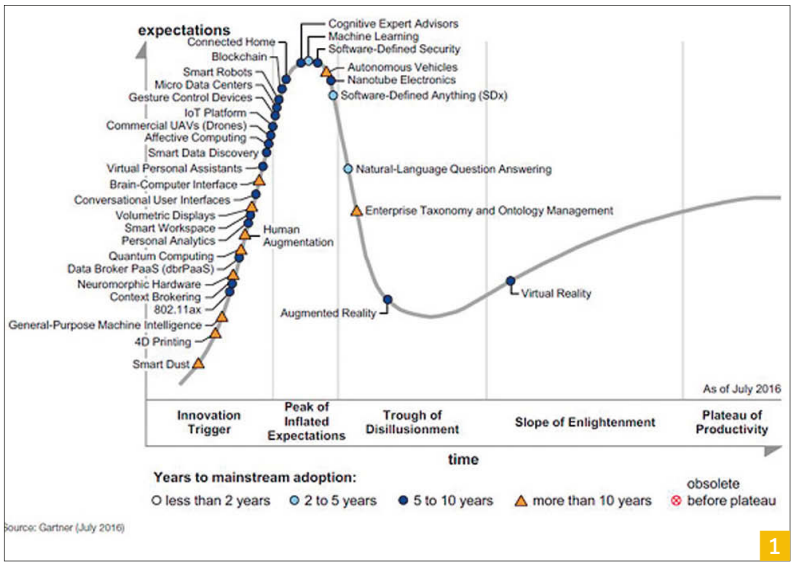
Les objets connectés : la clé du big data.

Qui dit « donnée » dit aussi « captation des données » or l'installation des capteurs n'est pas une étape simple : la mise en place du réseau électrique et du réseau de communication est coûteuse et doit répondre à de nombreuses contraintes liées au milieu industriel. **L'évolution des objets connectés en termes d'autonomie et de connectivité permet de résoudre ces complexités.**

La blockchain : au-delà du buzzword

La blockchain est une technologie à l'origine du Bitcoin qui pour rappel, cumule 45 milliards de dollars de valeur en 8 ans d'existence, sans aucune attaque réussie. La blockchain assure l'authenticité de la monnaie virtuelle et empêche sa double utilisation ou la réception de monnaie sans valeur lors d'échanges de crypto monnaies.

Ces deux technologies, objets connectés et blockchain, se retrouvent à la pointe de la courbe de la hype de Gartner : 1



1

ALORS, EST-IL POSSIBLE D'INTÉGRER LA BLOCKCHAIN AU SEIN D'OBJETS CONNECTÉS POUR LES INDUSTRIES ?

Débutons par une courte explication de ce qu'est la blockchain. Il s'agit d'un registre de transactions public et partagé entre tous les utilisateurs d'une même blockchain. Chaque transaction est signée et permet de savoir qui a initié chaque transaction. A chaque insertion, l'ensemble des participants valide la transaction et, si plus de 50% des nœuds l'acceptent, cette dernière est définitivement inscrite dans le registre. **L'évolution d'une blockchain se fait donc à la majorité.**

Comme tout registre, l'historique ne doit pas être modifiable. Il existe plusieurs manières de verrouiller l'historique des 19 transactions. Les plus connues sont la « Preuve de Travail » et la « Preuve d'Enjeu ». L'important étant simplement de savoir que l'historique de la blockchain est infalsifiable : **cette assurance offre une confiance qui permet de se passer d'une tierce partie entre deux participants.**

Intégration de la blockchain au flux de données

Notre première piste consiste à intégrer la blockchain à notre chaîne de traitement des données. Notre architecture IoT est la suivante :

Objet à [Passerelle] à IBM Watson IoT contenant un serveur MQTT et NodeRed à Maximo

La blockchain apportant un gage de confiance lors de transactions, la garantie maximale implique d'intégrer la blockchain au plus près de la captation de la donnée sur l'objet connecté en lui-même et non sur une passerelle ou un cloud dédié aux IoTs. Ainsi la donnée produite par l'objet sera authentifiée dès sa création.

De ce fait, cette authentification peut être faite bien plus simplement que par la

blockchain. La signature électronique joue ce rôle depuis longtemps et fait d'ores-et-déjà partie de notre vie quotidienne (dans nos cartes bancaires, tout simplement). **Une solution d'IAM (*Identity and Access Management*) serait donc tout à fait apte à authentifier nos objets et leurs données.**

La blockchain à tout prix

Mais alors, comment utiliser cette fameuse blockchain dans nos IoTs ?
Notre première piste n'était visiblement pas la bonne car déjà résolue par des solutions informatiques existantes. La problématique à laquelle répond la blockchain est de l'ordre de la relation entre différentes parties. Aussi, Sajida ZOUARHI, cofondatrice de Blockchain France indique dans un article sur Medium : « Rappelez-vous, faire une blockchain tout seul revient à utiliser une base de données ».
Pour trouver le cas d'usage idéal, nous avons identifié différents cas.

Cas du fournisseur d'objet connecté

Le fournisseur d'objet connecté peut-il être considéré comme une des parties des transactions ? N'étant pas acteur de la logique business de l'entreprise, il n'a pas de raison d'avoir accès à nos données.
Cas exceptionnel : le fournisseur d'IoT souhaite donner accès aux données de ses objets sous forme d'abonnement. Une utilisation de la blockchain est alors envisageable avec comme bénéfice pour l'abonné d'être certain que les données n'ont pas été modifiées avant de lui être transmises. Cependant, là encore, l'objet connecté pourrait lui-même modifier une donnée avant de la mettre sur la blockchain.

Diviser pour mieux intégrer la blockchain

Une autre solution pour transformer notre système en un système multipartite est de nous diviser en sous-parties fonctionnant entre elles. Etant plusieurs à interagir, la blockchain pourrait donc être intégrée, non ? En divisant nos objets par type de capteur (thermomètre, accéléromètre, gyroscope…), on appliquerait alors le mécanisme de consensus pour valider des valeurs envoyées par les capteurs. Mais comment différencier une valeur correcte d'une valeur anormale ? Et ce, lorsqu'un des objectifs des objets connectés est justement de détecter des anomalies ? Impossible. De nouveau, il n'y a pas de cas d'usage de blockchain lié aux IoTs dans ce contexte de remontée de donnée. Malgré nos efforts, il est réellement nécessaire d'avoir deux entités séparées qui interagissent entre elles pour que la blockchain dévoile son intérêt et permette d'ajouter de la confiance en la données partagée entre les parties.
La fondation Linux l'a bien compris et a lancé le projet Hyperledger.

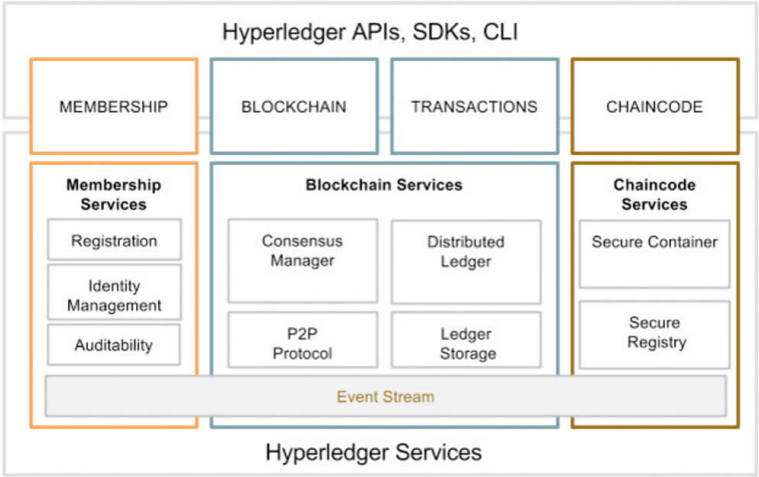


Figure 2: Hyperledger reference architecture

HYPERLEDGER FABRIC

Le sous-projet ***Fabric***, financé entre autres par IBM, propose une blockchain novatrice dédiée aux entreprises. Son fonctionnement consiste à établir des contrats entre deux ou plusieurs participants d'un même chanel. Chaque transaction doit être validée par *n* participants parmi *m* participants validateurs. Ces paramètres sont définis à la création d'un contrat et, une fois cette transaction validée, un « *Ordering Service* » se charge de répartir l'information à toute partie concernée par la transaction en veillant à ce que chaque individu du réseau n'ait accès qu'à l'information le concernant.
Voici une courte présentation :
https://youtu.be/js3ZjxbO8TM

HYPERLEDGER À PETITE ÉCHELLE : SIMULACRE DE BLOCKCHAIN À BASE DE BDD

A l'échelle de deux participants, il est possible de reproduire le fonctionnement d'*Hyperledger* avec un système de gestion de base de données.
Cela requiert une base de données en réplication avec un serveur placé chez chacun des participants. Dans cette base de données, seule l'insertion de transactions est autorisée. A chaque insertion en base, la donnée est signée par les deux participants afin de prouver la prise en compte.
Une table d'audit non répliquée doit ensuite être mise en place par chaque participant afin de rendre la base de données infalsifiable. Ainsi, si une des deux parties décide de supprimer une information, l'autre partie pourra prouver l'existence d'une donnée désormais disparue.
Cette solution permet d'utiliser des systèmes connus, stables et fiables dans le but de simuler le fonctionnement d'une technologie encore jeune et complexe, pas nécessairement adaptée à une mise en place en production à l'heure actuelle.
N'oublions pas que, malgré ses 8 ans d'existence, la blockchain est encore une technologie nouvelle dont l'utilisation en entreprise est encore limitée par une compréhension partielle de ses apports.
Par le biais de cet article, nous espérons vous avoir aidé à éclaircir ce que la blockchain n'est pas, à travers sa faible utilité dans le cas d'usage de la remontée de données dans un SI unique et maitrisé.
La blockchain pourrait cependant avoir un apport sur différentes applications dans le secteur de l'industrie à travers *Hyperledger Fabric*, qui mériterait un article à part entière. •

L'Intelligence Artificielle et le deus ex machina

L'intelligence Artificielle (IA) alimente tous les fantasmes et délires, et ce constat ne va pas aller en s'améliorant. Car finalement, l'IA est aussi bien un objet réel que l'on relie volontairement à l'ordinateur et aux robots qu'un objet mystique de Science-Fiction. Ce dernier point a suscité de nombreuses évolutions et souvent sombres comme Dark City / Matrix, Asimov ou K. Dick, Caprica / Battlestar Galactica, 2001, Alien, Humans, Real Humans. Jules Verne avait imaginé des machines autonomes. Et bien entendu, n'oublions pas Karel Capek et la fameuse pièce de théâtre R.U.R. et la première utilisation du mot « robot ».

La rédaction

La machine pensante, l'automatisation, ne sont pas des concepts récents. Raymond Lulle (1232 – 1315) imagina l'Ars Magna, une machine à penser rudimentaire pour répondre aux questions philosophiques. Mais nous pourrions remonter à l'invention de la machine d'Antycithère ou encore à Archimède, un des grands penseurs scientifiques de l'Humanité.

En 1956, le General Problem Solver a l'ambition de démontrer des théorèmes mathématiques. Une fondation pour les systèmes experts et les calculs formels. Dix ans plus tard, le premier chatbot, assistant personnel, était développé : ELIZA. Son créateur était Joseph Weizenbaum, qui s'appuiera sur les travaux d'Alan Turing. ELIZA avait été conçu pour imiter un thérapeute avec des questions et un certain nombre de réponses. Le système reconnaissait des mots clés. La renaissance d'un mot clé permettait d'envoyer une réponse, etc. ELIZA inspira ensuite plusieurs jeux des années 1970.

L'IA ne souleva pas beaucoup d'intérêt et peu de passions du grand public avant les années 2010. La puissance des matériels, la baisse des coûts, le besoin d'interagir en temps réel et naturellement de la machine avec l'Homme, les évolutions technologiques, l'explosion des GAFAM, ont permis le renouveau de l'IA en confiant le Machine Learning, le Deep Learning, le Big Data, à des processeurs spécifiques. Les assistants personnels poussent l'IA au cœur de notre quotidien. Les réseaux neuronaux, les algorithmes génétiques ne sont qu'une infime partie de cette mouvance.

C'est en 1956 que le terme IA apparaît et devient une science, notamment grâce à McCarty qui utilise pour la première fois le terme. Une des idées de base du chercheur était la possibilité de comprendre et de décrire l'intelligence humaine, donc de pouvoir la recréer artificiellement. Même si tout le monde n'était pas d'accord avec la vision de McCarty et de ses collègues.

Les années 1970 marquent le véritable tournant dans l'IA. Avec comme dogme de départ : les programmes doivent avoir une connaissance approfondie du sujet, de la réponse. De là se pose la question de la connaissance, comment la récupérer, la traiter, l'utiliser.

Donner une définition « simple » de l'IA n'est pas possible, car aujourd'hui, si vous posez la question à 10 personnes, vous aurez sans doute 10 réponses différentes même si des éléments communs se retrouveront d'une réponse à une autre.

Vous entendrez souvent parler de deux types d'IA : l'IA forte, l'IA faible. L'IA forte peut être comprise comme un système global pour reproduire et traiter le plus possible de fonctions d'un cerveau et dans les capacités d'actions et de réflexions. On peut citer la voiture autonome, la robotique, les systèmes cognitifs. L'IA faible est une ambition fonctionnelle réduite et agit sur un domaine réduit. Elle ne se voit pas ou très faiblement.

« L'intelligence artificielle est la science dont le but est de faire faire par une machine des tâches que l'homme accomplit en utilisant son intelligence. »

« Étude des activités intellectuelles de l'homme pour lesquelles aucune méthode n'est a priori connue. L'informatique est la science du traitement de l'information, l'IA s'intéresse à tous les cas où ce traitement ne peut être ramené à une méthode simple, précise, algorithmique»

(d'après J.L Laurière).

François Tonic

Questions – réponses avec **Brice Cornet**

Fondateur de Simple CRM, Brice est un observateur de la technologie. L'IA n'est pas une inconnue pour lui. Questions – réponses.

Comment tu définis l'IA ?
Tu commences directement par la question qui fâche ! La définition de l'IA est vraiment une question qui divise actuellement. Certains mettent cette division sur le côté jeune de la technologie, or il n'y a rien de plus faux. On peut situer la naissance de l'IA en 1943, quand McCulloch pose la définition des neurones formels. Peu de personnes le savent, mais IBM sortait déjà en 1959 des logiciels d'IA consacrés à la géométrie et qui tournaient sur des IBM 704 à lampes. Donc l'IA est aussi vieille que l'informatique. Par contre, cette branche de l'informatique a été mise de côté pendant des décennies car elle est très gourmande en puissance de calcul. C'est cette mise de côté qui explique cette sensation de jeunesse et cette dissension en termes de définition. Dissension qui est fortement amplifiée par le « IA washing » actuel, un « IA washing » souvent porté par une presse qui n'y comprend absolument rien mais qui trouve que le terme IA fait vendre. Bref, pour répondre à ta question, je vais poser une définition qui est le reflet de mon point de vue personnel et qui sera sans doute critiqué par d'autres acteurs, puisque le sujet fait tant débat.
L'IA est, pour moi, un ensemble de technologies permettant de résoudre des problèmes complexes, résolution qui passe par une imitation du comportement humain. Dans ces technologies, tu retrouves une série de grandes familles qui sont le machine learning, le natural language processing (NDLR - exemple : la traduction d'une langue), le speech (NDLR - la lecture d'un texte par une voix synthétique), les systèmes experts (NDLR - exemple : système d'aide à la décision), la robotique et la reconnaissance son et image.

Aujourd'hui quels sont les usages les plus courants pour l'IA ? Et quels marchés / utilisateurs ils visent réellement ?
Pour répondre à ta question, je pense qu'il faudrait différencier deux choses : les volumes d'utilisations de l'IA, qui sont actuellement très orientés B2C et « *besoin du quotidien* », des tendances IA à venir, car ce qui monopolise actuellement l'actualité des marchés, ce sont les tendances d'utilisations de demain, c'est-à-dire les entreprises.
Avant, IA pour le B2B rimait avec forte consommation de processeur et donc coût d'utilisation non rentable. Les GPU ont changé la donne, et, conjugués au volume astronomique de données disponibles, ils permettent à l'IA de faire son retour en force dans le secteur des entreprises. Il y a des mines de données à ciel ouvert non ou mal exploitées : l'IA est donc le nouvel eldorado. Donc un grand marché de l'IA pour les 5 prochaines années sera celui des systèmes experts qui aideront à l'exploitation du Big Data.
A l'instant « T », c'est pourtant l'utilisation de l'IA par Monsieur et Madame *ToutLeMonde* qui domine. Les deux usages les plus courants sont le machine learning puisqu'il équipe tous les systèmes anti-spam qui nous facilitent la vie depuis une grosse dizaine d'année, et la reconnaissance image qui est présente dans nos appareils photos et nos téléphones, via l'identification d'un visage sur le capteur photographique. On doit ensuite ajouter le natural language processing utilisé par tous les systèmes de reconnaissance vocale dans le monde.
Enfin, tu peux ajouter une grosse interaction avec les systèmes de machine learning qui traitent du big data, ne serait-ce que par l'énorme volume de données traitées chaque jour par les moteurs de recherche.
Honnêtement, je n'ai jamais vu de statistiques réelles, mais en prenant en compte ces différents grands segments, on doit couvrir 98% de la consommation IA journalière.

On parle beaucoup de Machine Learning, Deep Learning, de processeurs dédiés, de modèles d'apprentissage, d'algorithme génétiques, l'IA est une combinaison de différentes couches finalement ?
Comme toujours : « *l'informatique, un métier mais surtout des métiers* » !
Si la création d'un site web fait appel à un administrateur système, un codeur, un graphiste et un gestionnaire de base de données… inutile de te faire un dessin quand on parle d'IA.
L'IA est en fait un maillage de technologies et avec parfois peu de code derrière, mais plutôt une utilisation intelligente de système existants interconnectés.
On peut fort bien imaginer un système IA, équipé de capteurs contenant eux-mêmes de l'IA préconfigurée afin de capter uniquement ce qui est réellement important. Ceux-ci remontent ensuite les

informations dans un TensorFlow, lui-même exploitant un jeu d'analyse préexistant, qui renvoie ensuite dans un système expert fait maison sur une base de bibliothèque en C++ dédié à la programmation par contrainte. Je pense sincèrement qu'à terme, le plus gros du travail de l'IA sera d'être capable de définir l'architecture la plus performante, ce qui sous-entend qu'il va falloir des architectes qui auront une vue à 360 degrés sur les possibilités des outils du marché.
Pour l'anecdote, chez Simple CRM, on a perdu deux ans de dev à cause d'un mauvais choix d'architecture. Nous avions confié la réalisation du moteur de détection de client potentiel à une entreprise externe, spécialisée dans cette problématique. Le souci est que cette entreprise s'est enlisée dans une architecture beaucoup trop lourde. Après deux ans de développement, nous avions une IA qui nous produisait des clients potentiels à 2,1 € du coût de production. En mettant à la poubelle ce travail, en étudiant les solutions, en cadrant les besoins réels, en sortant certains traitements de cercles IA pour les replacer dans un bon vieux traitement SQL qui réinjectait ensuite dans un flux IA, on est passé à un coût de production de 0,0000001 € du client potentiel. Tu comprends donc mieux toute l'importance de poser les bons choix, et tu comprendras aussi qu'au final, l'IA est une branche de l'informatique comme les autres où le succès de ton projet passe tout simplement par une analyse carrée de tes besoins et des solutions techniques disponibles.

Pour un développeur, quel est le plus difficile à coder quand on parle IA ?
De mon point de vue, c'est la notion de machine learning, surtout si elle est supportée par un réseau de neurones. Au collège, j'étais un cancre en mathématiques, mais par contre, quand j'ai commencé à coder, j'étais vraiment bon. Un jour, un professeur m'a demandé si j'avais suivi un cursus littéraire classique au collègue de type « *latin – grec* », ce qui était en effet le cas. Ce professeur m'a dit que cela se voyait dans ma façon de coder mais que cela n'était nullement un handicap, mais tout simplement une façon de penser et un style. Cette approche plutôt littéraire que mathématique ne m'a jamais posé de problème dans mon métier, sauf lorsque l'on a pris la direction de l'IA. Cette branche de la programmation repose vraiment sur une compréhension fine de certains concepts mathématiques qui personnellement m'échappent parfois totalement !

Dans ton cas, tu proposes un assistant complet. Quels ont été les obstacles techniques ? Quel budget représente un tel développement ?
Le premier obstacle, c'était la disponibilité du savoir et des outils, puisque l'on a attaqué très tôt le sujet. Le cahier des charges du projet date de 2014 et actuellement, nous avons réalisé +/- 60% de l'IA de notre assistante virtuelle.
Résultat : comme tout pionnier, on a beaucoup expérimenté et l'on s'est beaucoup trompés. Nous avons posé pas mal de mauvais choix techniques en termes d'architecture, comme expliqué précédemment. On a aussi posé de mauvais choix managériaux. Enfin, soyons honnête : mea culpa, j'ai posé de mauvais choix car quand le projet IA est arrivé chez Simple CRM, il y a eu un vent de panique chez les développeurs. Cette science jeune, vue comme un territoire vierge, faisait peur. J'ai donc posé le choix de sous-traiter alors que j'aurais dû poser le choix de créer progressivement une culture de l'IA dans l'entreprise, avec une montée en puissance des compétences.
En y réfléchissant, le premier obstacle n'est pas technique mais humain. C'est l'incompréhension même que soulève le terme IA, avec ses mythes et légendes qui n'aident vraiment pas le boulot côté RH. Depuis la fin de l'année 2017, on a commencé à construire une programme d'éducation à l'IA pour tous les employés, y compris les gens du marketing ou de l'administratif, histoire que le sujet soit réellement compris.
Ensuite, on a eu le manque d'outil. Cela a été assez vite réglé avec l'arrivée d'environnement comme TensorFlow, Scikit Learn ou DSSTNE. Mais là on a été confrontés à un autre souci : que choisir sachant que si le projet qui soutient cet environnement s'arrête, nous allons nous retrouver avec une architecture rapidement obsolète.
En termes de budget, je ne peux pas être trop limpide là-dessus pour des questions de confidentialité. Je te dirai juste que l'on a brûlé entre 250 000 et 500 000 €.

Demain, les profils techniques IA seront-ils obligatoires ?
Oui et non. Non car des industries logicielles continueront sans doute à vivre sans IA pendant encore quelques années. Oui car l'IA va devenir à terme un outil du quotidien comme les autres qui facilitera aussi le travail des codeurs.
Comme toute branche de l'informatique, l'IA va s'affiner et s'hyper spécialiser, ce qui va demander bien entendu des profils de plus en plus pointus.
Nous n'en sommes qu'au début de cette nouvelle révolution industrielle. Je pense que l'on va voir arriver à terme des besoins en architecture ASIC (NDLR : application specific integrated circuit) construits sur mesure par rapport à des besoins spécifiques.
On aura aussi des boîtiers genre *Raspberry*, dédiés à l'IA, qui permettront des récoltes d'informations spécifiques et des traitements précis qui remonteront ensuite sur des moteurs IA plus importants.
On aura ces fameux architectes IA qui devront poser le choix délicat des bonnes solutions à interconnecter.
Je pense que l'IA ouvre également une nouvelle porte, à des profils de codeur différents. On aura besoin sans doute de personnes qui auront étudié la biologie, pour mieux comprendre et affiner la notion de déplacement chez les robots.
On aura besoin de psychologues pour les outils IA de gestion des ressources humaines. On aura besoin de financiers pour les IA dédiées aux systèmes décisionnels.
Bref, les profils des experts IA vont être beaucoup plus multiples et ouverts que ne le sont ceux de l'informatique *classique*, ce qui va entraîner une mutation intéressante de nos environnements professionnels mais aussi et surtout humains ! •

David Carteau
carteau.david@gmail.com

L'apprentissage incrémental à population Partie 1

L'apprentissage incrémental à population, en anglais « Population Based Incremental Learning » (PBIL), a été proposé en 1994 par Shumeet Baluja [1]. Inspirée des algorithmes génétiques, cette méthode est présentée par son auteur comme étant à la fois plus simple à implémenter que ces derniers, tout en donnant des résultats plus rapides et plus précis.

L'apprentissage incrémental à population fait partie de la famille des algorithmes à estimation de distribution [2]. Il s'agit d'une métaheuristique utilisée, comme les algorithmes génétiques, pour résoudre des problèmes d'optimisation difficile. La différence principale entre les deux techniques réside dans le fait qu'ici, les caractéristiques génétiques des individus constituant une population ne sont pas issues de mécanismes de sélections et de croisements mais d'une distribution de probabilité. C'est cette distribution de probabilité qui évolue au fil des générations, et non les individus eux-mêmes, pour converger vers une solution optimale. La simplicité de la méthode vient du fait que chaque gène se voit affecter d'une probabilité d'être actif ou non. La création d'une population d'individus est ainsi extrêmement aisée : il suffit de générer aléatoirement des individus en tenant compte de la probabilité d'activation de chaque gène. Au lancement de l'algorithme, chaque gène, représenté par un bit (0 ou 1), se voit affecté d'une probabilité d'activation de 50 % et cette probabilité va évoluer au fil des générations, en fonction de l'adaptation des individus à leur environnement, pour tendre soit vers 100 % (tous les individus générés auront ce gène activé) soit vers 0 % (tous les individus générés auront ce gène désactivé).

Pseudo-code

Voici le pseudo-code de l'algorithme :

```
Initialiser le vecteur de probabilité (probabilité d'activation de chaque gène = 50%)

Boucler tant que le critère d'arrêt n'est pas rempli

    Créer  une population aléatoire à partir du vecteur de probabilité

    Évaluer chaque individu de la population (mesurer son adaptation ou « fitness »)

    Mettre à jour le vecteur de probabilité

Fin de la boucle
```

Création du vecteur de probabilité

Passons maintenant à l'implémentation en Java : créons d'abord une classe « Vector » représentant la distribution de probabilité. C'est le cœur de l'algorithme. Le constructeur de la classe prend un seul paramètre : le nombre de gènes représentant un individu. A l'initialisation d'une nouvelle distribution, chaque gène a une probabilité d'activation qui vaut 50 % : pour représenter ces probabilités, le type Java « float » est suffisant en termes de précision (nombre flottant codé sur 32 bits) :

```
public class Vector {
    private float[] probability;

    public Vector(int size) {
        probability = new float[size];

        for (int i = 0; i < size; i++) {
            probability[i] = 0.5f;
        }
    }

    public float getProbability(int i) {
        return probability[i];
    }
}
```

Mise à jour du vecteur de probabilité

La mise à jour du vecteur de probabilité est très simple. Elle nécessite d'avoir identifié à l'issue de chaque itération l'individu le mieux adapté et celui le moins adapté à son environnement. La mise à jour repose alors sur deux principes : l'application d'un taux d'apprentissage qui va faire évoluer plus ou moins rapidement la probabilité d'activation associée à chaque gène, et l'application de mutations génétiques aléatoires visant à éviter une convergence trop rapide vers un extremum local.

a) la probabilité d'activation associée à chaque gène va d'abord évoluer selon la formule suivante (où $gène_{i,meilleur}$ représente 0 ou 1, la valeur du $gène_i$ de l'individu le mieux adapté) :

$$probabilité(gène_i) = probabilité(gène_i) * (1{,}0 - taux) + (gène_{i,meilleur}) * taux$$

Dans cette formule, le taux d'apprentissage (« *learning rate* ») va contrôler directement la vitesse de convergence de l'algorithme : plus il sera élevé, plus l'algorithme convergera vite vers une solution, mais pas nécessairement optimale. Au contraire, plus il sera faible, plus les chances de converger vers un optimum global (et non local) seront grandes. La probabilité d'activation de chaque gène va naturellement varier en fonction de la valeur du gène du meilleur individu (elle va décroître si le gène vaut 0, et inversement).

Dans sa publication, Shumeet Baluja montre qu'il est possible d'améliorer la convergence de l'algorithme en complétant la notion de taux d'apprentissage par celle de taux d'apprentissage

« négatif » (« *negative learning rate* »). Cette amélioration consiste à faire varier la probabilité associée à un gène plus fortement dans le cas où le gène porté par l'individu le mieux adapté est de valeur opposée au gène porté par l'individu le moins adapté.
L'exemple suivant illustre le processus de mise à jour des probabilités : partant de la distribution de probabilités {0.5, 0.5, 0.5, 0.5} et des individus '0111' (« meilleur » individu) et '0001' (individu le moins adapté), une nouvelle distribution de probabilité possible serait {0.45, 0.60, 0.60, 0.55}. Les gènes n°1 et n°4 étant identiques entre les deux individus, la probabilité de chacun de ces gènes va varier en ne tenant compte que du « learning rate » (ici fixé à 10%). A contrario, les gènes n°2 et n°3 étant différents entre les deux individus, ce taux va être complété du « negative learning rate » (ici également fixé à 10%), résultant ainsi dans une variation plus importante (+ 20 %) des probabilités associées.

b) la probabilité d'activation associée à chaque gène va ensuite subir (ou non) une mutation.
A la différence des algorithmes génétiques classiques, où les mutations frappent directement le génome des individus en modifiant la valeur d'un ou plusieurs gènes, les mutations vont s'appliquer ici à la distribution de probabilité. Selon une probabilité de mutation définie, appelée « *mutation probability* », une ou plusieurs probabilités de la distribution se verront affecter d'une variation infime (« *mutation shift* ») dont le sens sera aléatoire (+ / -). Cette modification va donc perdurer durablement au fil des générations, comme une mutation est transmise à la descendance d'un individu dans les algorithmes génétiques classiques.
Complétons maintenant la classe « Vector » pour ajouter la méthode de mise à jour de la distribution de probabilité :

```
public void update(Individual winner, Individual looser) {
  int size = probability.length;

  // mise à jour des probabilité
  for (int i = 0; i < size; i++) {
    float rate = Constants.LEARN_RATE;

    if (looser.isBitSet(i) != winner.isBitSet(i)) {
      rate += Constants.NEGATIVE_LEARN_RATE;
    }

    probability[i] = probability[i] * (1.0f - rate) + (winner.isBitSet(i) ? 1.0f : 0.0f) * rate;
  }

  // mutations
  for (int i = 0; i < size; i++) {
    if (Rand.getFloat() < Constants.MUTATION_PROBABILITY) {
      String info = "Mutation bit " + i + ", old = " + probability[i];

      probability[i] *= 1.0f - Constants.MUTATION_SHIFT;
      if (Rand.getFloat() < 0.5f) {
        probability[i] += Constants.MUTATION_SHIFT;
      }

      System.out.println(info + ", new = " + probability[i]);
    }
  }
}
```

Pour éviter de surcharger le code, les taux sont implémentés ici sous forme de constantes, mais il est bien sûr préférable de les rendre paramétrables. C'est en effet en les faisant varier qu'il sera possible d'ajuster la vitesse de convergence et la précision des résultats en fonction du problème à résoudre.

Critère d'arrêt et convergence globale

Le pseudo-code présenté précédemment ne précise pas le critère d'arrêt de l'algorithme. Dans sa publication, Shumeet Baluja utilise un critère « classique » qui est de dérouler l'algorithme pendant un certain nombre d'itérations. Il est cependant possible d'aller plus loin et de calculer le taux de convergence global de l'algorithme et de stopper ce dernier si le taux dépasse un certain seuil (par exemple 99%).
Pour calculer ce taux de convergence globale, il faut se souvenir que chaque probabilité va lentement converger au fil des générations vers 0 % ou 100 %. A 0 %, chaque nouvel individu généré verra le gène correspondant désactivé. A 100 %, le gène sera systématiquement activé. Il est possible de calculer à chaque instant la convergence de chaque probabilité, et d'en faire la moyenne pour obtenir la convergence globale :

$$convergence = \frac{1}{n} * \sum_{1}^{n} (2 * |probabilité(gène_i) - 0{,}5|)$$

Complétons la classe « Vector » pour y ajouter la méthode correspondante :

```
public float getConvergence() {
  int size = probability.length;

  float convergence = 0.0f;

  for (int i = 0; i < size; i++) {
    convergence += 2.0f * Math.abs(probability[i] - 0.5f);
  }

  convergence /= (float) size;

  System.out.println("Convergence = " + convergence);

  return convergence;
}
```

Création et évaluation des individus

Le lecteur attentif aura remarqué dans les extraits de code ci-dessus l'usage de classes Java non encore définies : « Rand » et « Individual ». La classe « Rand » sert simplement à obtenir un nouveau nombre aléatoire de type « float ». Elle offre également la possibilité de fixer simplement d'une exécution à l'autre les valeurs aléatoires retournées, ce qui utile lors des phases de débogage :

```
public class Rand {
```

```
    // remplacer par "random = new Random(0)" pour retourner d'une exécution à l'autre les
mêmes valeurs aléatoires
    private static Random random = new Random();

    public static float getFloat() {
        return random.nextFloat();
    }

}
```

La classe « Individual » va quant à elle être bien plus intéressante. Commençons par créer un nouvel individu à partir d'une distribution de probabilité et d'un nom fournis en paramètres :

```
public class Individual {

    private String name;
    private String genome;

    public Individual(Vector vector, String name) {
        int size = vector.getSize();

        StringBuilder sb = new StringBuilder();

        for (int i = 0; i < size; i++) {
            if (Rand.getFloat() < vector.getProbability(i)) {
                sb.append("1");
            } else {
                sb.append("0");
            }
        }

        this.name = name;
        this.genome = sb.toString();
    }

    public boolean isBitSet(int i) {
        return (genome.charAt(i) == '1');
    }

    public String getName() {
        return name;
    }

    public String getGenome() {
        return genome;
    }

}
```

L'évaluation de l'adaptation d'un individu, également appelée « fitness », va permettre d'évaluer la qualité d'une solution au problème recherché. Cette évaluation est fortement dépendante du problème posé, mais doit être d'autant plus grande que la solution se rapproche de la solution optimale. Prenons le problème « classique » du sac à dos. Un randonneur dispose d'un sac à dos et d'un ensemble d'objets à sa disposition, qu'il peut décider d'emporter avec lui pour partir en randonnée. Chaque objet a un poids et une utilité donnés. Le sac à dos étant malheureusement de taille limitée, le randonneur va donc devoir opérer un choix et n'emmener qu'une sélection d'objets. Il va aussi chercher à maximiser l'utilité des objets emportés. Cette utilité totale peut servir ici de « fitness ».

Complétons la classe « Individual » pour calculer cette valeur. Ici, l'activation d'un gène (valeur du gène fixée à '1') signifiera que l'objet correspondant au gène est déposé dans le sac à dos. Si l'ensemble des objets déposés dans le sac excède le poids transportable, la « fitness » sera positionnée à -1, signifiant que le génome (la solution) n'est pas viable :

```
private static Map<String, Integer> cache = new HashMap<>();

public void evaluate() {
    if (cache.get(genome) == null) {
        int weight = 0;
        int utility = 0;

        for (int i = 0; i < genome.length(); i++) {
            if (genome.charAt(i) == '1') {
                weight += Constants.WEIGHT[i];
                utility += Constants.UTILITY[i];
            }
        }

        if (weight > Constants.MAX_WEIGHT) {
            utility = -1; // solution non valide
        }

        cache.put(genome, utility);
    }
}

public int getFitness() {
    return cache.get(genome);
}
```

Là encore, pour des besoins de simplicité de code, les valeurs utiles sont regroupées dans une classe nommée « Constants ». Elles devraient être idéalement paramétrables. La lecture du code montre également que les évaluations sont mises en cache. C'est une petite amélioration qui permet de gagner énormément de temps lors des dernières itérations où les populations générées contiendront – avec une probabilité croissante – des individus déjà évalués par le passé. •

Liens

[1] http://www.ri.cmu.edu/pub_files/pub1/baluja_shumeet_1994_2/baluja_shumeet_1994_2.pdf
[2] https://fr.wikipedia.org/wiki/Algorithme_à_estimation_de_distribution

Alexandre Variengien,
étudiant en MPSI au lycée Champollion à Grenoble, lauréat du prix Bernard-Novelli 2017
alexandre.variengien@gmail.com

Comment j'ai créé une IA avec un algorithme génétique

Durant mon année de terminale, dans le cadre de la spécialité ISN pour le bac, j'ai développé un projet en Python. Il s'agit d'un jeu de dominos avec la possibilité - la partie la plus intéressante - d'affronter une intelligence artificielle (IA) qui a appris toute seule une stratégie optimale. En étant assez satisfait, j'ai donc ensuite présenté ce projet au prix Bernard Novelli que j'ai eu la chance de remporter.

J'ai choisi de développer ce jeu de dominos car il révèle une partie très importante de stratégie. Malgré l'image qui lui est souvent attribuée de jeu auquel on joue lorsqu'on est enfant, arriver à créer des blocages pour les autres joueurs tout en gardant une diversité de chiffres dans son jeu et en minimisant le nombre de points qu'il nous reste est loin d'être trivial.

Avant de rentrer dans les détails techniques du projet, d'abord quelques précisions sur les règles que j'ai utilisées : 1

Le jeu de domino est un jeu de société se jouant de 2 à 4 joueurs. Il se joue avec des tuiles, communément le jeu de double 6 comme ci-contre. Au début du jeu chaque joueur pioche. Lors du déroulement du jeu, chaque joueur joue à tour de rôle en posant un domino. Pour cela il faut qu'un des chiffres d'une tuile de son jeu soit identique à un des cotés du jeu présent sur le plateau. Puis il pose le domino pour que les chiffres identiques soient côte à côte. S'il ne peut pas jouer, il pioche jusqu'à ce que le domino pioché puisse être joué. Si la pioche est vide, il passe son tour. La manche se termine lorsqu'un joueur a posé tous ses dominos ou bien lorsque tous les joueurs ont passé leur tour. A la fin de la manche, on compte le nombre de points présents dans le jeu de chaque joueur et on l'ajoute au nombre de points qu'il a déjà. Le jeu se finit lorsque la limite de points définie au début du jeu est atteinte, souvent de 50 ou 100 points. Le gagnant est celui qui a le moins de points à la fin de la partie.

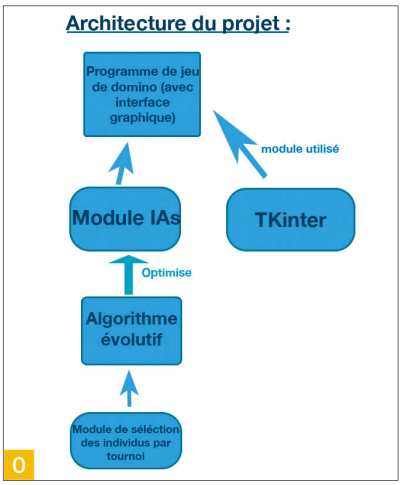
Et quelques améliorations aussi ajoutées :

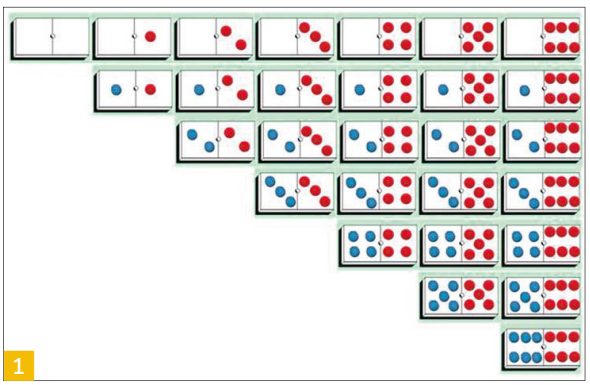
- possibilité de jouer avec un jeu double 9 plutôt que double 6 ;
- mode de jeu "collaboration" : l'IA en retard au score partage ses informations avec le joueur lorsqu'il n'est pas en tête.

Architecture technique

Le projet a été codé en Python en environ 2500 lignes, et j'utilise le module TKinter pour l'interface graphique. Le projet se divise en 4 grandes parties : un module IA, le programme principal qui fait tourner le jeu et qui contient un ensemble de fonctions appelées pour l'affichage et l'interaction avec le joueur, un algorithme évolutif d'optimisation des IAs et un module de sélection des IAs dans l'algorithme évolutif. L'architecture du projet est présentée ci-contre et l'interaction avec le joueur. 0

Il y a plusieurs niveaux d'IA (facile, moyenne, difficile). Dans le module, chaque IA est modélisée par une fonction prenant en entrée le jeu du joueur qu'elle incarne, la situation du jeu, et donne en sortie le domino du jeu à jouer si elle le peut ou l'ordre de piocher. Pour le développement en lui-même, j'étais en binôme avec Jérémie Lavault qui s'est chargé de la partie graphique alors que je créais tout le jeu et l'IA. Durant la phase de coding j'ai dû faire face à différents pièges. Comme par exemple avec les objets listes - que j'ai beaucoup utilisés - qui ont en Python un traitement assez spécial : Python utilise des pointeurs pour lier une variable à la case mémoire. Lorsqu'on crée une nouvelle variable qui a la valeur d'une liste déjà existante, Python, partisan du moindre effort, ne crée qu'un nouveau pointeur de la nouvelle variable vers la case mémoire déjà existante sans copier son contenu ! Si bien que lorsqu'on modifie la valeur de la liste ainsi créée, on écrit dans la case mémoire et on modifie alors les deux listes en même temps. Pour pallier cela, on peut utiliser la copie forcée qui copie complètement tous les éléments de la première liste vers la seconde. Ce phénomène peut causer un certain nombre de bugs frustrants, inexplicables si on l'ignore (ce qui a été le cas durant le développement ce projet). 2 3





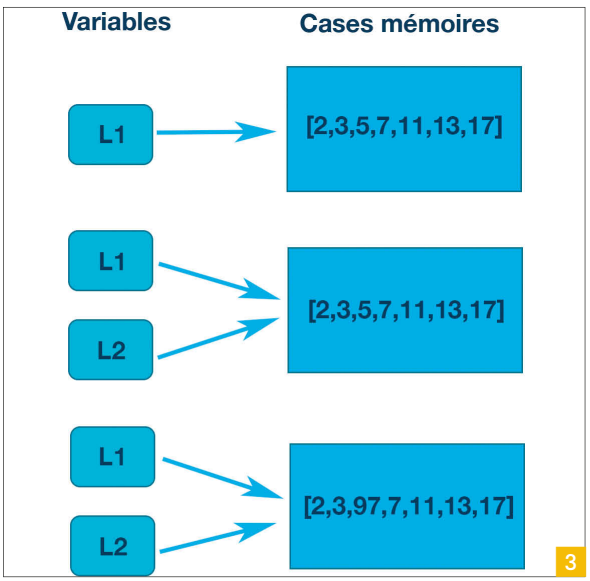

```
>>> L1 = [2,3,5,7,11,13,17]
>>> L2 = L1
>>> L2[2] = 97
>>> L2
[2, 3, 97, 7, 11, 13, 17]
>>> L1
[2, 3, 97, 7, 11, 13, 17]
```

Copie forcée

```
>>> L1 = [2,3,5,7,11,13,17]
>>> L2 = L1[:]
>>> L2[2] = 97
>>> L2
[2, 3, 97, 7, 11, 13, 17]
>>> L1
[2, 3, 5, 7, 11, 13, 17]
```

2

Copie naïve



Copie naïve d'une liste du point de vue des variable

Le développement de l'IA

Dans le jeu de dominos, chaque joueur n'a accès qu'à un nombre limité d'informations : son jeu et le plateau, pour ce qui est de savoir ce que possèdent les autres (très utile pour anticiper les mouvements), on ne peut s'appuyer que sur des déductions. Pour donner les moyens à l'IA de prendre une décision, c'est à dire choisir le domino le plus judicieux à jouer dans cette situation, j'ai créé, grâce à mon expérience de jeu, différentes fonctions d'évaluation des dominos du jeu de l'IA. C'est à dire qu'à chaque tuile jouable on associe un score et l'IA joue le domino possédant le plus gros score.

```
276     domi_joue = chercher_meilleur(domi_scores)
```

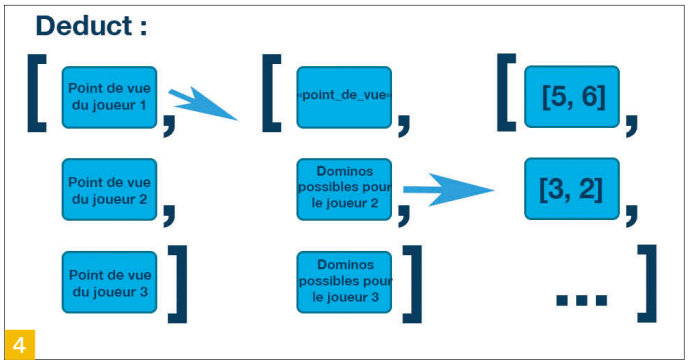
Ce score est la somme pondérée (le choix des coefficients sera explicité ensuite) des scores des différentes fonctions d'évaluation.

```
271     score = score_div * coef_div + score_double * coef_do + score_valeur * coef_val + pts_bloq
```

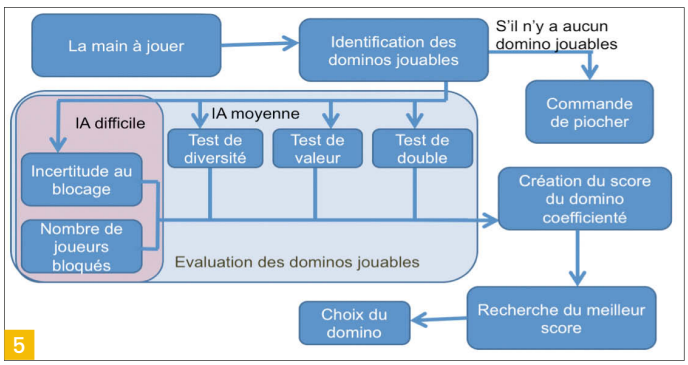
Le but est de regarder chaque domino de l'IA sous le plus d'angles possibles afin de prendre la décision la plus avisée.
Elles sont les suivantes :

- "test_doubles" qui renvoie simplement 1 si le domino est un double et 0 sinon.
- "test_valeur" qui renvoie la valeur du domino, le joueur gagnant étant celui qui possède le moins de points à son compteur général, jouer le plus vite possible les dominos à grande valeur doit être pris en compte par l'IA.
- La fonction "test_diversite" qui prend en paramètre le domino bien sûr mais aussi la main de l'IA, et la position du chiffre qui ne sera pas collé au jeu dans le domino à tester, si c'est celui-ci qui est retenu. La fonction renvoie le nombre d'occurrences de ce chiffre non collé au jeu dans la main du joueur, ainsi cela permet de préserver une diversité des chiffres que possède l'IA dans son jeu pour éviter de se faire bloquer trop facilement. Par exemple si l'IA doit jouer sur du 5 et elle ne peut poser que les dominos "5 - 4" ou "5 - 2", si elle a plusieurs fois le chiffre "2" autre part dans son jeu, de plus si "5 - 4" est son seul domino qui a du "4" alors elle a tout intérêt à jouer le "5 - 2" pour éviter d'être bloquée au 4 par la suite.

De plus j'ai aussi implémenté un système de déduction des jeux des autres joueurs, cela rend l'IA très performante puisse qu'elle peut prédire un coup à l'avance le jeu et alors créer des blocages et à coup sûr faire piocher les autres. Je me suis inspiré de la gymnastique mentale qui est nécessaire quand on y joue pour l'automatiser et rendre l'IA encore plus performante. Mais pour pouvoir arriver à déduire le jeu des autres joueurs, il faut analyser les actions qu'ils ont faites.
Par exemple lorsqu'un joueur pioche cela signifie que dans son jeu il ne possède aucun domino comportant les chiffres composant les deux bouts du jeu sur lequel il a pioché. On peut alors retenir ces deux chiffres, on est certain que si on a plus tard la possibilité de faire en sorte que les deux bouts du jeu soient ceux-ci il faut absolument le faire !
Plus techniquement, pour arriver à automatiser ce système de déduction le plus généralement possible, j'ai mis en place une matrice à plusieurs dimensions de tous les dominos que peuvent posséder chaque joueur du point de vue de chaque joueur. Je m'explique : le jeu de tous les joueurs est différent, de plus tous les dominos étant uniques, chaque joueur peu être sûr qu'aucune de ses tuiles ne se retrouvera dans un jeu adverse, on utilise alors cela pour faire des déductions. Cela explique pourquoi le point de vue qu'on prend doit être pris en compte dans cette matrice "deduct". Lorsque l'on tombe sur les déductions du joueur 1 sur le joueur 1 par exemple, cela ne sert à rien de le remplir, les jeux des joueurs étant stockés sur une autre liste, alors on a juste une chaîne de caractères "point_de_vue" pour ne pas décaler les indices de la liste. 4
Et alors, toutes les IA ont accès à ce tableau depuis leur point de vue. Cela



4



5

permet de savoir facilement quels chiffres bloquent à coup sûr quels joueurs. En particulier, ce qui est utile pour l'IA, c'est de savoir quel chiffre bloque le joueur qui va jouer à sa suite, cela permettra de le faire piocher à coup sûr si elle a les moyens de les mettre en bouts de jeu. C'est dans ce but que j'ai développé la quatrième fonction d'évaluation "test_blocage" :
Pour chaque élément du jeu de l'IA elle calcule le nombre de dominos bloqués des déductions si ce domino est joué. Si le nombre de dominos bloqués est inférieur à la longueur du jeu du joueur cible, alors c'est certain qu'il possédera un domino non bloqué, il pourra alors le jouer et la tentative de blocage tombera à plat, on ne prend pas ça en compte. Seulement si on trouve un blocage possible on sélectionne le moins incertain et on crée le score de blocage comme ceci :

```
265     pts_bloq = score_blocage[0] * coef_bloq + score_blocage[1] * coef_incert
```

où l'incertitude au blocage est la différence entre le nombre de dominos bloqués dans les déductions de l'IA et le nombre de dominos que possède le joueur ciblé (le coefficient appliqué à l'incertitude est négatif) et le score du blocage vaut 1 si un blocage a été trouvé et zéro sinon.
De plus, dans le but de pouvoir moduler le niveau de l'IA à affronter, j'ai créé différents niveaux à partir de ces modèles : l'IA basique fait un choix purement aléatoire parmi ses dominos jouables, l'IA moyenne ne prend en compte que les fonctions d'évaluation de double, valeur et diversité pour faire son choix, tandis que l'IA difficile les utilise toutes.
Finalement, voici le fonctionnement de l'IA depuis l'instant où elle reçoit sa main jusqu'à l'action qu'elle ordonne d'exécuter : d'abord identifier les dominos jouables ou piocher, puis l'évaluation et enfin la recherche du meilleur domino. 5
Et ensuite, voici le code Python correspondant. Il comporte le traitement de certains cas particuliers : quand le domino est le premier à être joué, la liste "B" qui contient les deux bouts du jeu en temps normal est initialisée à [-5,-5] de manière à prendre ce critère en compte. Tous les dominos sont jouables au premier tour. De plus, certains dominos sont jouables des deux cotés du jeu, (par exemple 5-4 lorsque les bouts du jeu sont 5 et 4) alors on les traite comme deux dominos séparés car ils auront des résultats aux fonctions d'évaluation différents. De plus, la fonction "tester_pion" permet de déterminer si un domi-

no est jouable ou non et s'il l'est, de quel coté du jeu. Cette information est ensuite liée au domino pour pouvoir être ensuite utilisée pour son évaluation via une liste à deux éléments. 6 7

Détermination des coefficients grâce à un algorithme évolutif

Cependant, on peut facilement se rendre compte que tous les scores renvoyés par les fonctions d'évaluation ne sont pas à prendre avec la même valeur. Par exemple un 1 renvoyé par la fonction "test_valeur" n'a pas du tout la même valeur dans la décision que le 1 renvoyé par la fonction "test_double" : la valeur n'intervient dans le choix du domino qu'en dernier pour départager deux dominos plus ou moins semblables alors qu'il est capital de se débarrasser de ses doubles (ils portent les deux mêmes chiffres ce qui augmente les chances d'être bloqué ensuite). C'est pour cela qu'il est nécessaire de pondérer ces scores.

Mais pour trouver des coefficients, c'est impossible de les déterminer vraiment précisément à la main, en se basant uniquement sur de vagues intuitions. De plus il est clairement impossible de tous les tester : il y a 6 variables à déterminer, même en se restreignant à l'intervalle [0,10] avec 5 pas de 0.1 il faudrait tester 10 milliards de possibilités, ce qui est clairement infaisable.

C'est pour cela que j'ai mis en place un algorithme dit évolutif ou génétique. C'est à dire, un algorithme utilisant les mécanismes de l'évolution de Darwin par sélection naturelle. Son fonctionnement peut être illustré par une boucle comme le montre le graphique ci dessous : 8

Ici un individu est une IA, où plus précisément, une méthode de jeu qui utilise le principe de décision décrit précédemment et avec des coefficients définis. Alors ces individus ont pour ADN un ensemble de 6 nombres (5 coefficients à optimiser et 1 indice du mutativité qui définit l'amplitude de la mutation lors de la reproduction).

On crée d'abord une population de 100 individus avec des ADNs choisis aléatoirement.

Puis dans cette population de départ, on sélectionne les individus qui sont les meilleurs au jeu de dominos. C'est l'étape la plus importante de cet algorithme, c'est ici qu'on va définir le trait que les individus vont devoir optimiser, ce qu'on recherche. Dans notre cas, l'IA est destinée à être affrontée par le joueur, il faut donc qu'elle soit la plus compétitive possible, c'est à dire capable de développer une stratégie qui fonctionne même si son opposant est lui aussi stratège. Une idée naturelle pour créer un test qui sélectionne cette qualité est d'organiser un tournoi entre tous les individus de la génération et ensuite sélectionner ceux qui ont fini dans les premiers. J'ai alors créé une fonction qui organise un tournoi, avec pour but qu'en le moins de matchs possible, le classement soit le plus représentatif de la capacité des individus. Au premier tour les couples d'adversaires sont déterminés aléatoirement puis on les classe en fonction de leurs victoires. Ensuite pour chaque tour, pour chaque participant, on choisit l'adversaire le plus proche de lui dans le classement contre lequel il n'a jamais joué. Et pour s'assurer de l'objectivité des matchs, le gagnant d'une confrontation est celui qui a gagné le plus de parties de dominos sur les 100 disputées.

Une fois que la crème de notre génération a été identifiée, on lui donne le droit de se reproduire pour que la génération suivante hérite de ses capacités, et on introduit aussi dans la copie, une petite variation, une mutation aléatoire. C'est ici l'autre étape capitale de ce programme : les mutations aléatoires dont l'amplitude est codée par l'ADN du parent, par son indice de mutabilité, comme ci dessous.

```
for pos in range(len(methodes_jeu)):  # on reproduit les methodes sélectionnées
    reprod = []
    for i in range(long_meth):
        reprod.append(methodes_jeu[pos][i] * (1 + uniform(-methodes_jeu[pos][long_meth - 1], methodes_jeu[pos][long_meth - 1])))
```

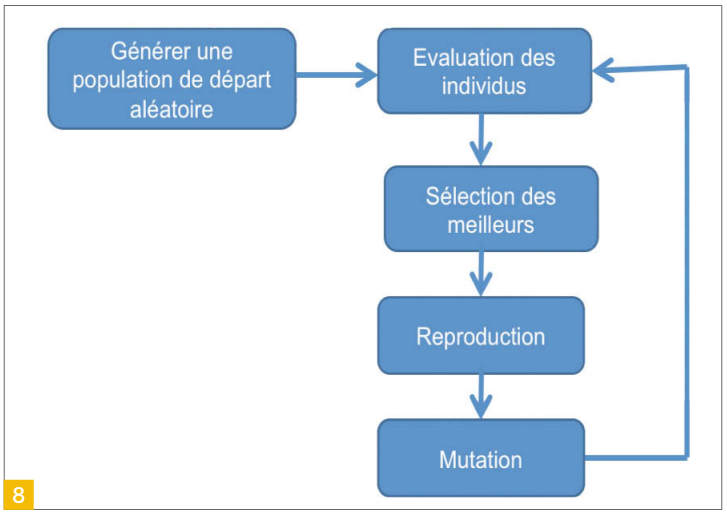
Ce sont elles qui vont permettre l'exploration de nouvelles stratégies qui ne seront pas présentes dans la génération des parents. On obtient comme ça une

```
    domi_scores = []

    for elem in pions_jou:
        score_div = test_diversite(elem[0], mon_jeu,elem[1][1])  # on créé les scores venant des différentes fonctions d'évaluation
        score_double = test_doubles(elem[0])
        score_valeur = test_valeur(elem[0])
        score_blocage = test_blocage(elem[0], elem[1], mes_deduct, les_jeux, B, tour_IA)

        pts_bloq = score_blocage[0] * coef_bloq + score_blocage[1] * coef_incert  # le coefficient d'incertitude est negatif, si le
                                                                                  # score créé par la fonction d'évaluation du blocage est
                                                                                  # négatif, il ne faut pas que ca impacte les autres scores
        if pts_bloq < 0:
            pts_bloq = 0
        score = score_div * coef_div + score_double * coef_do + score_valeur * coef_val + pts_bloq

        domi_scores.append([elem,score])  # on créé une liste de toutes les informations sur les dominos jouables avec leur score associé

    domi_joue = chercher_meilleur(domi_scores)  # on cherche le domino avec le meilleur score : c'est celui ci que l'ia va jouer

    if len(domi_joue) == 3: #si le domino est ambidextre
        return [domi_joue[0], domi_joue[1][0]]  #on renvoit aussi le coté du jeu sur lequel il doit être joué
    else:
        return domi_joue[0]
```

6

```
def IA_difficile(mon_jeu, B, mes_deduct, les_jeux, tour_IA, coef_div, coef_do, coef_val, coef_bloq, coef_incert):
    """fonction qui renvoie l'action à executer avec ces bouts de jeu et ces dominos en main : soit le domino à jouer, soit "pioche" avec une prise de décision qui prend en compte la main actuelle de l'IA, les propriété du seul domino
    et les les jeux des adversaires par déductions, les mains adverses pour évaluer leurs longueurs et le tour de l'IA"""
    global pions_jou
    global domi_scores

    pions_jou = []

    if B[0] == -5:
        for domino in mon_jeu:
            pions_jou.append([domino, ["slot_inutile_car",
                                       "premier_domino"]])  # on créé un resultat de test spécial pour le premier domnio
    else:
        for domino in mon_jeu:
            if test_ambidextre(domino,B):  # si le domino peut être posé des deux cotés du jeu, on créé deux dominos jouables possibles pour
                # le meme domino mais ils auront des résultats au test de diversité et de blocage différents
                # ils sont donc traités comme deux dominos séparés indépendants
                for k in range(2):
                    test = tester_pion(domino, B,k)  # on définit le côté du jeu sur lequel on veut jouer le domino : un pour chaque coté
                    pions_jou.append([domino, test,"ambidextre"])  # on appelle ambidextre un domino qui peut se placer des deux cotés du jeu,
                                                                   # on le signale dans l'identification du domino

            else:
                test = tester_pion(domino, B)
                if test != False:  # si le pion est jouable
                    pions_jou.append([domino,test])  # on créé la liste des dominos jouables avec leur coté collé associé,utile pour le test de diversité
    if len(pions_jou) == 0:
        return "pioche"
```

7



8

nouvelle génération. Et on revient à l'étape de sélection pour parcourir la boucle autant de fois qu'il est nécessaire pour obtenir des résultats satisfaisants.

Même si la première génération complètement aléatoire sera mauvaise, une fois qu'on aura sélectionné les meilleurs - ou plutôt les moins pire ici - et introduit des mutations, on pourra ensuite répéter sélection-reproduction-mutation. Si un individu à la chance de bénéficier, grâce à une mutation, du moindre petit changement dans le bon sens vers des coefficients qui permettent de prendre des décisions plus avisées, alors il sera sélectionné, copié et encore optimisé pour se rapprocher encore plus d'une stratégie optimale via ses enfants, petits-enfants arrières- petits-enfants, etc. Et pour la plupart, ils subissent une mutation aléatoire qui réduit leurs compétitivités, alors ils seront éliminés. Et à la fin, en laissant le programme tourner assez longtemps (≈ 1h pour moi) on obtient une IA qui se défend vraiment bien ! En effet, elle fait preuve d'une vraie stratégie car elle bat à plate couture l'IA qui prend des décisions aléa-

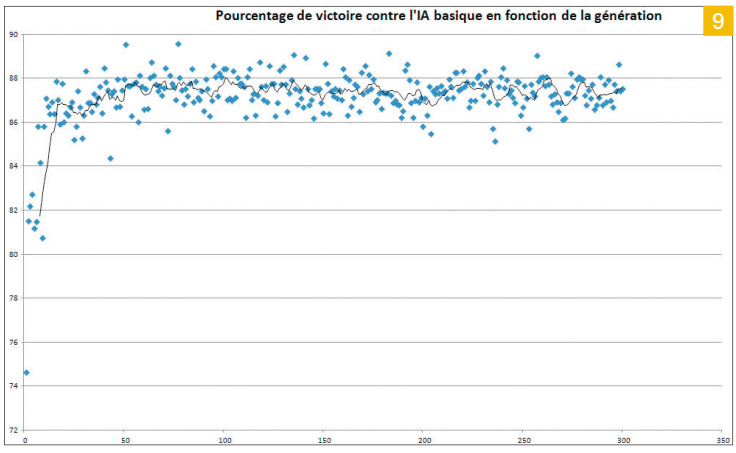
toires parmi ses dominos jouables : 88 à 12 sur une moyenne de 10 000 parties. Ce mécanisme utilisé est le même que le moteur de l'évolution à l'origine de toutes les espèces. Seulement, dans la nature, des individus qui possèdent des caractères codés par leurs molécules d'ADN interagissent dans un environnement physique et on laisse agir les milliers d'années pour obtenir une espèce parfaitement adaptée à son milieu. Ici des objets, des variables et une sélection virtuelle modélise une population et une pression évolutive et on laisse agir la puissance de calcul du processeur pour obtenir une stratégie optimale à un problème. Cette technique, à la croisée de la biologie et de l'informatique, nous donne une solution à un problème qui paraissait impossible à résoudre avec la force brute.

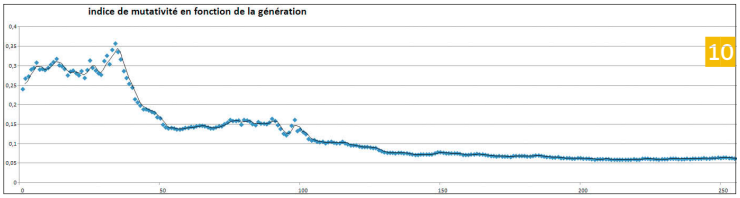
De plus cet algorithme nous livre des informations très intéressantes quand on analyse les données de l'évolution. 9

D'abord si on regarde les performances du meilleur représentant de chaque génération contre l'IA jouant aléatoirement (IA basique) on observe clairement une stabilisation rapide vers un maximum, ce qui est bien ce qu'on attendait : la convergence vers une stratégie optimale. Ce qui confirme cela, c'est aussi l'évolution de l'indice de mutabilité qui décroit au court des générations : lorsqu'on est arrivé à une stabilisation, tous les individus qui s'en éloignent sont moins performants et sont éliminés, alors les individus donnant des enfants les plus proches d'eux sont sélectionnés. 10

Et enfin, on peut aussi interpréter la stratégie qu'a trouvé l'algorithme. L'ADN final est : 1.00; 0.3662; 0.1123; 19.6215; 0.7253; 0.0556. (coefficients de : doubles, diversité, valeur, blocage, incertitude au blocage, indice de mutativité). Il colle à l'intuition que j'avais évoquée au début entre la valeur et le double : il accorde 10 fois plus d'importance au critère du double qu'à un point de valeur. Et on voit aussi que le système de déduction est très largement exploité par l'IA puisque le plus gros coefficient est appliqué à la fonction d'évaluation des blocages. Cela est aussi confirmé par la baisse de performance si on ne prend pas en considération les déductions (c'est comme cela que joue l'IA moyenne) on obtient seulement 79% de victoire contre l'IA basique.

Et surtout, une fois en face à face contre elle, elle ne fait pas de quartier et la vaincre nécessite une grande stratégie.





Le lien du projet avec les problématiques actuelles en IA

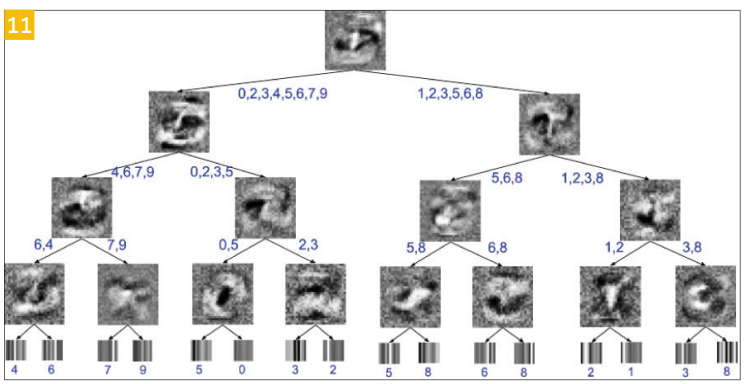
Outre l'aspect ludique de ce projet, il permet d'illustrer les défis auxquels les chercheurs en IA font face actuellement. D'abord, sur l'efficacité de l'algorithme génétique. Même si il est actuellement largement dépassé en termes de performance par les réseaux de neurones artificiels, sa grande flexibilité dans les objets qu'il traite lui permet de trouver un autre usage. En effet il est aussi utilisé pour résoudre des problèmes très complexes comme l'optimisation de l'architecture de réseaux de neurones de reconnaissance d'images (Hanxiao Liu et al.), structure qui était jusqu'à maintenant choisie arbitrairement, en suivant une intuition, par tâtonnements. C'est une tâche à laquelle il est parfaitement adapté : la forme des individus de la population peut être très différente tant que l'évaluation est possible. Alors que les réseaux de neurones ont eux une structure rigide et ont besoin d'une taille fixe pour les données qu'ils manipulent et le résultat qu'ils donnent.

Un autre des défis actuels de l'IA est d'arriver à comprendre de quelle manière elle arrive à prendre des décisions. Particulièrement avec les réseaux de neurones, on les entraîne avec un grand ensemble de données et ensuite, ils généralisent leur entraînement à des données qu'ils n'avaient jamais vues auparavant. Cependant ils créent un effet "boîte noire" : la plupart de leurs décisions sont correctes mais on n'a aucune idée de pourquoi ils ont pris tel choix et non un autre tant le niveau d'abstraction de ces objets est grand. Une des solutions apportées est de faire en sorte que ces réseaux génèrent des paramètres de décisions optimisés et interprétables plutôt que la décision elle-même. C'est ce que j'ai fait dans mon projet : les résultats donnés par l'algorithme évolutif sont les coefficients que l'IA applique aux différentes fonctions, ce qui est facilement interprétable. Si on demande

-"Pourquoi tu as favorisé ce choix plutôt qu'un autre ?"

-"Parce qu'il permettait de bloquer le joueur suivant au 4 et 3, je le sais car il a pioché sur du 4 il y a 6 tours, et tous les 3 sont sur le plateau et cela tout en gardant 2 dominos contenant 4 à côté dans mon jeu " peut par exemple répondre l'IA. Une explication détaillée est toujours disponible. 11

C'est aussi ce qu'a fait un groupe de chercheurs de l'équipe Google Brain sur des problèmes plus complexes ("Distilling a Neural Network Into a Soft Decision Tree",Nicholas Frosst et al.) : ils ont entraîné un réseau de neurones à créer un arbre de décision facilement interprétable et performant pour classifier des images de chiffres écrits à la main en fonction du chiffre écrit grâce à une succession de filtres comme illustré ci-contre. Ce genre d'avancée permettra une meilleure interprétabilité de nos algorithmes lorsqu'elle est nécessaire. Cette compréhension aidera aussi sûrement à apporter une solution au problème de la responsabilité des erreurs commises par ces systèmes qui peuvent être lourdes de conséquences dans certains domaines comme avec les voitures autonomes. •



Vincent Perrin
Expert IA, IBM France

Un chatbot ? Oui ! Mais après ?

L'intelligence artificielle était sur toutes les lèvres en 2017, dans de très nombreux articles et conférences. Le chatbot a été la meilleure illustration de l'apport de l'intelligence artificielle comme catalyseur d'une profonde transformation de nos interactions avec ou via la « machine ». A l'aide de l'IA, un chatbot casse les silos entre le marketing, les ventes et le service client en changeant drastiquement la manière dont un client cherche, compare ou achète un produit. Mais, la mise en place d'un chatbot n'est que le début d'une aventure comme l'expose l'étude de Hubspot.

En 2017, de nombreux chatbots ont vu le jour en France pour servir différents buts pour les entreprises : augmentation des ventes, amélioration de la relation client, voire même buzz marketing dans certains cas, avec plus ou moins de succès. 2018 va voir cette nouvelle interaction se développer de manière exponentielle.
Je souhaite ici partager quelques éléments de réflexion pour mettre un chatbot en œuvre ou le faire évoluer :

- Les différents types de chatbots ;
- Les canaux à intégrer ;
- Quelques bonnes pratiques.

Types de chatbots

Je vois à ce jour 4 types de chatbots déployables permettant d'avoir une expérience utilisateur plus ou moins prononcée.

TYPE 1 : Chatbot "*basique*"

La plupart des chatbots mis actuellement en œuvre sont basés sur le triptyque « Intentions/Entités/Dialogue ». La stratégie de raisonnement est basée sur la meilleure compréhension/identification du contenu et du contexte saisi par l'utilisateur.

« Comprendre au mieux la question pour fournir la réponse pour laquelle le chatbot a été entraîné. »

Cette fonctionnalité, proposée par la plupart des sociétés et des startups, nécessite un entraînement autour des thèmes définis à l'avance et en nombre limité. Plus le nombre d'intentions est important, plus le processus d'apprentissage est long et complexe, et cela pour deux raisons, il faut entraîner le système à reconnaître toutes ces intentions et gérer la désambiguïsation.

TYPE 2 : Chatbot ++

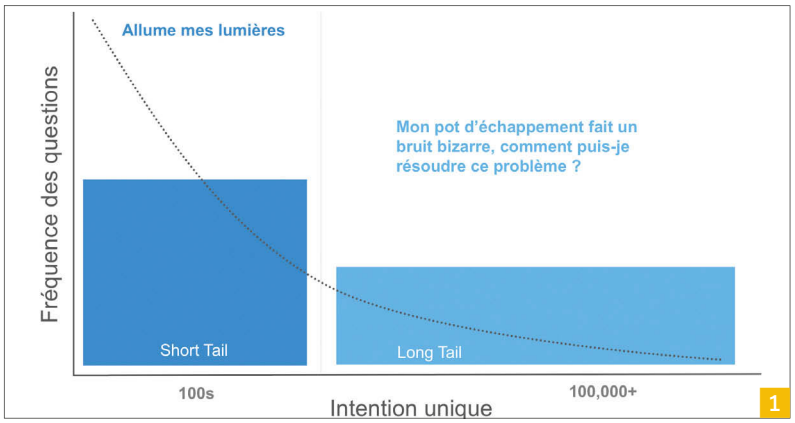
Ce type de chatbot est une extension du précédent implémentant des services tiers pour améliorer l'expérience. Par exemple, par l'analyse du ton permettant d'adapter la forme ou le fond d'une réponse fournie par le chatbot, ou bien par de la reconnaissance visuelle, dans le cas du support client pour faciliter l'identification du produit (version) à supporter.

TYPE 3 : Chatbot « *longue traîne* » (Long Tail)

Dans de nombreux cas, l'utilisateur peut poser une question ne faisant pas (encore) partie des thématiques prévues, non imaginée par les concepteurs du chatbot ou peu fréquente, trop complexe à reconnaître.
Pour traiter tous ces cas, IBM propose un service complémentaire nommé Watson Discovery (https://www.ibm.com/watson/services/discovery) qui a une stratégie de raisonnement permettant de trouver la réponse la plus appropriée à la question dans un corpus documentaire. 1
Ce type de chatbot permet d'adresser de manière plus simple un nombre beaucoup plus grand de questions utilisateurs et donc d'augmenter la satisfaction relative à l'usage du chatbot. Il est aussi indispensable lorsque le sujet du chatbot est très large ou complexe (par exemple répondre à des questions autour du Code des impôts) ou bien dans le cas d'un support employé (utilisation B2E), par exemple agent d'un call-center, support produit…
Autre exemple l'application mobile « Ask Mercedes » qui permet de remplacer le manuel de la voiture : https://www.ibm.com/blogs/think/2017/11/end-of-the-car-manual/

TYPE 4 : Chatbot applicatif (Back-office, Décisionnel, Moteur de recommandation…)

Le chatbot travaille sur le langage naturel pour supporter l'interaction humaine, mais doit s'appuyer sur des informations ou des processus applicatifs tiers.
Par exemple, dans le cas d'un chatbot bancaire, la technologie permet de reconnaître le besoin (par exemple « Quel est le solde de mon compte ? »), mais nécessite de s'appuyer sur le back-office bancaire pour



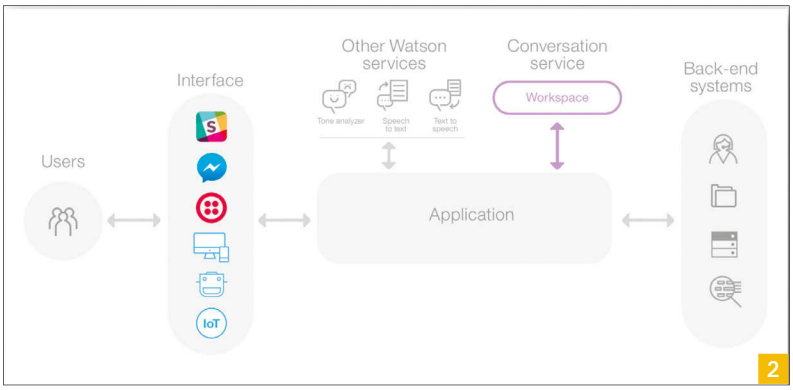
1

récupérer cette information et la fournir à l'utilisateur. Le schéma ci-dessous illustre bien cette approche : 2

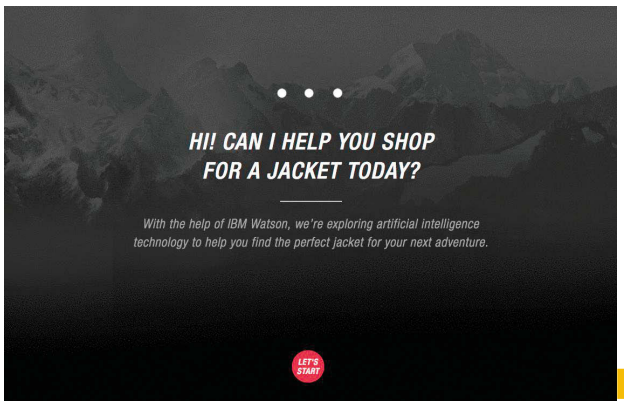
Dans le cas d'un chatbot e-commerce comme The North Face, cette technologie permet déjà de réduire le temps pour trouver et acheter le bon produit. Au lieu de regarder une longue liste de produits, le consommateur peut expliquer en langage naturel son besoin par exemple, "un manteau pour aller faire du ski en février à Chamonix" et le chatbot lui retourne une liste limitée de suggestions. 3

Ou bien, dans le cas d'un chatbot Telco pour permettre de proposer le meilleur service/contrat en fonction des besoins exprimés par le consommateur et alignés avec des contraintes légales/contractuelles définies via un moteur de règles tel que IBM ODM (Operational Decision Manager).

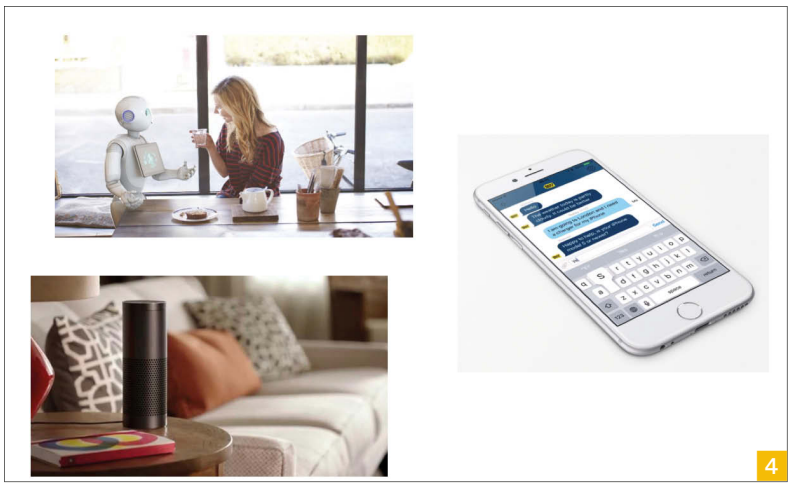
Exemple d'utilisation : https://github.com/ibm-cloud-architecture/refarch-cognitive-prod-recommendations



2



3


4

Note

Dans une récente évolution (décembre 2017), l'appel à un système tiers peut se faire directement depuis le service IBM Watson Conversation, au travers d'IBM Cloud Functions. (https://developer.ibm.com/recipes/tutorials/watson-conversation-ibm-cloud-functions-to-do-a-longtail-chatbot/)

Canal ou canaux d'interaction

Dans le cadre d'un usage orienté expérience client, le chatbot représente la marque et interagit en son nom. Dans une expérience qui se veut de plus en plus multi, voire omni-canal, le chatbot doit être exploité sur de multiples canaux tels que Web, Applications mobiles, messageries de groupe, robots et assistants vocaux (article sur l'explosion de l'interaction vocale). 4

Une marque doit s'adresser à ses clients-là où ils se trouvent. La technologie de chatbot doit être agnostique du canal et permettre d'en couvrir le plus grand nombre et de s'adapter aux spécificités de ceux-ci.

Une organisation ne doit avoir qu'un seul et unique moteur conversationnel. Il serait, en effet, dommageable d'avoir à réaliser plusieurs entraînements et d'avoir des réponses différentes suivant les canaux.

Bonnes pratiques

Voici quelques bonnes pratiques issues de projets réalisés sur le marché :

- Un chatbot oui, mais pourquoi ? Question clé. Vous devez définir des objectifs clairs pour votre chatbot et partager les capacités du chatbot avec les utilisateurs ;
- Restez raisonnable, ne sur-promettez pas les capacités du chatbot. Le chatbot, puisqu'il représente la marque, ne doit pas dégrader l'expérience client par des objectifs peu clairs ou hors d'atteinte. (Utilisation d'un chatbot de Type 3) ;
- Un chatbot, via quel canal ? paragraphe ci-dessus sur les canaux d'interaction ;
- Quid de l'escalade ? Dans tous les cas, le chatbot ne pourra traiter 100% des demandes, il faut donc définir une stratégie d'escalade (transfert à un agent humain en temps réel, réponse par email à J+1…) ;
- Équipe-projet ? Ne sous-estimez pas l'entraînement, mais surtout le réapprentissage (suivi). Après la mise en œuvre d'un chatbot, il n'est pas « encore » le moment de partir en vacances, il faut analyser les interactions avec celui-ci et réaliser un réapprentissage pour lui permettre de mieux reconnaître des intentions (anciennes ou nouvelles) ;
- Recherchez la technologie et l'intégrateur capables de vous accompagner dans la vision, dans l'industrialisation et dans l'intégration ;
- Prenez en main la technologie et mettez en œuvre un centre de compétences pour vous rendre autonome ;
- Appliquez une méthode agile et itérez dans la mise en œuvre du chatbot avec à l'esprit l'approche « Fail Fast » - délivrer plus vite en ajustant en continu.

Note

Cette liste n'est pas exhaustive et doit s'adapter à chaque projet.

cellenza
DOESITBETTER | Conseil - Expertise Microsoft & méthodes agiles

Nicolas ROBERT
Technical Manager chez Cellenza
@NicoRobPro

Coline THOMAS
Consultante Data et AI chez Cellenza
@colineto

Services Cognitifs : l'IA sur étagère (et plus)

Depuis quelques mois, l'Intelligence Artificielle est devenue le sujet chaud de la communauté informatique et bien au-delà. Découvrons comment nous pouvons dorénavant exploiter la puissance des algorithmes en quelques lignes de code, sans délai, pour transformer nos applications et nos interactions avec les utilisateurs.

En ce début 2018, la réalité rattrape la fiction. Entre assistants virtuels, robots, voitures autonomes, les ergonomies changent et les interactions avec les utilisateurs se doivent d'être repensées. Oublions l'ergonomie dictée par la règle des 3 clics, l'heure est venue de proposer des expériences plus humaines, directes, profitant de tous les outils de notre quotidien qui ne se résument pas à une souris et un clavier.

Nous allons voir comment Microsoft propose une réponse via sa brique « Cognitive Services », proposant des services prêts à consommer, mais aussi des versions plus personnalisées qui vont révolutionner nos usages.

Services cognitifs, que cachent-ils ?

Vous l'aurez compris, l'idée de ces services est d'exposer de façon simple et rapidement accessible des « algorithmes intelligents pour **voir, écouter, énoncer, comprendre** et **interpréter** les besoins de vos utilisateurs au moyen de méthodes naturelles de communication ».

Microsoft propose un regroupement de ses services en 5 familles :

- « Vision », pour ceux liés à l'image et la vidéo ;
- « Microsoft Speech », pour le son ;
- « Langue » pour le texte ;
- « Connaissance » pour les bases de connaissances ;
- « Rechercher » pour les recherches à l'aide de Bing. 1

Ces familles exposent déjà plus d'une trentaine d'APIs. Pour ne citer que quelques APIs plus parlantes que d'autres, on y retrouve notamment la capacité de réaliser du Speech-to-Text, de la traduction à la volée, de la correction orthographique de texte, de la détection de visages sur des images, etc. Et cela en quelques clics !

A consommer à l'unité ou en combinaison

Les cas d'utilisations possibles sont très nombreux. On peut déjà illustrer l'intérêt des services cognitifs par des retours d'expériences existants : prenons le cas d'Uber qui, outre-Atlantique, utilise l'API « Face » pour vérifier l'identité de ses conducteurs. Cette vérification passe par la prise d'une photo via l'application qu'ils ont à disposition qui est ensuite comparée à une base existante de conducteurs et vérifie la correspondance.

Certains usages sont plus poussés en combinant différents services cognitifs au service d'une expérience utilisateur plus fluide et humanisée : c'est souvent le cas notamment dans le développement des robots de conversation, alias chatbots. Ces derniers sont de parfaites illustrations d'outils historiquement « peu intelligents » que l'on a pu enrichir au fil du temps avec l'arrivée de tels services. En effet, les premiers chatbots étaient généralement des outils n'étant en capacité de répondre qu'à des demandes très spécifiques, avec leur propre syntaxe. Que se passe-t-il lorsque l'utilisateur ne la connaît pas parfaitement ? On le perd.

Grâce au moteur de « Natural Language Processing » prénommé LUIS pour « Language Understanding Intelligent Service », on est maintenant en capacité de traiter automatiquement le langage naturel de l'utilisateur, sans avoir à lui demander de respecter une syntaxe inflexible. Il n'y a pas de magie derrière LUIS, cela nécessite bien évidemment la définition d'intentions, la fourniture d'exemples de phrases pour chacune de ces intentions, et de l'entraînement pour que les algorithmes soient ensuite capables de réaliser la bonne détection.

Pourtant, on reste bloqué par d'autres problématiques : comment peut-on gérer maintenant le fait que l'utilisateur fasse des coquilles

1

dans sa saisie ? La réponse passe peut-être par l'ajout de l'appel à l'API « Vérification orthographique Bing » en amont de l'analyse de LUIS, pour minimiser ces cas.

Et si vos utilisateurs écrivent en différentes langues ? Détectez la langue avec l'API « Analyse de texte », et si vous souhaitez conserver votre traitement en 1 seule langue, traduisez via l'API « Translator Text » !

Maintenant que l'on gère le texte, pourrait-on permettre à l'utilisateur de nous transmettre sa demande oralement en parallèle, afin de lui offrir plus de choix ? On va retrouver dans la famille « Speech » toutes les APIs nécessaires : l'API « Microsoft Speech Bing » pour la reconnaissance vocale, l'API « Translator Speech » pour la traduction. Vous pouvez même faire de la synthèse vocale pour répondre.

Un autre très bon exemple de combinaison de services cognitifs est l'outil « Video Indexer » de Microsoft disponible sur l'URL https://vi.microsoft.com/ : ce dernier permet d'indexer une vidéo, détecter les orateurs, identifier des mots-clés, faire de la détection de caractères sur les images (OCR), générer le transcript et d'autres fonctionnalités à l'aide de ces différents services.
A l'usage, on peut constater que l'outil est perfectible notamment sur les langues autres que l'anglais, ou sur les vidéos avec un enregistrement sonore de qualité moyenne, mais il reste bluffant pour une version préliminaire à laquelle on ne fait que fournir une simple vidéo.

Les voir en action

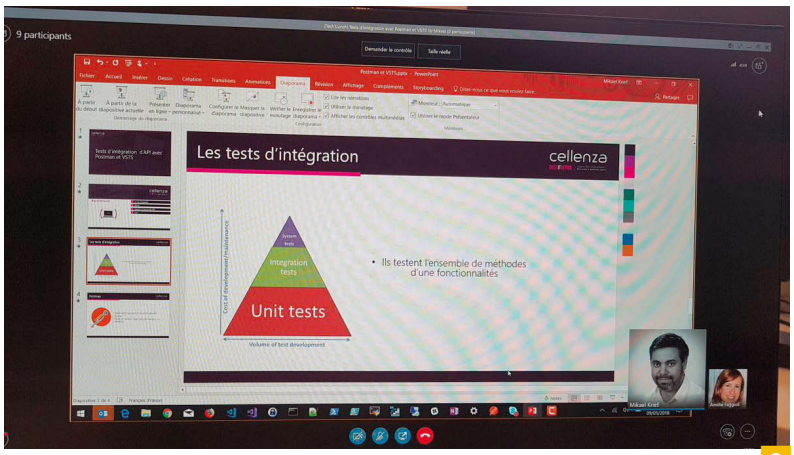
Afin de découvrir les services, il existe des démonstrateurs basiques sur la page https://azure.microsoft.com/fr-fr/services/cognitive-services/.
Il existe également une application « Intelligent Kiosk » disponible sur le Store Windows fournissant différentes démonstrations et dont les sources sont disponibles sur GitHub (https://github.com/Microsoft/Cognitive-Samples-IntelligentKiosk). Attention cependant sur les démonstrations liées à de la vidéo, elles épuisent très rapidement les quotas des clés d'API gratuites dont nous parlons ci-dessous !

Comment les utiliser ?

Pour aller un peu plus loin que ces démonstrateurs, la majorité des services (si ce n'est pas la totalité) sont accessibles à l'aide de clés d'API gratuites ! Vous pouvez les demander directement sur le lien https://azure.microsoft.com/fr-fr/try/cognitive-services/ . Pour d'autres produits comme LUIS, il faut simplement se connecter sur le portail du produit et une clé basique vous est automatiquement assignée.
Evidemment, ces clés gratuites ont des quotas d'utilisation mais ils permettent de se donner déjà une bonne idée de leur utilité sur des projets.

Une fois votre clé en main, tout va dépendre de votre profil de développeur et de votre projet. Vous pouvez notamment :

- Appeler directement les API via votre code. Ces dernières sont exposées à l'aide d'Azure API Management : https://westus.dev.cognitive.microsoft.com/docs/services
- Utiliser des librairies clientes le cas échéant : pour du développement .Net notamment, utiliser des packages NuGet notamment ceux du compte « ProjectOxfordSDK » (Oxford étant le nom de projet des Cognitive Services avant leur sortie officielle).



2

La documentation de chaque API fournit généralement des exemples d'utilisation dans différents langages (C#, Java, JavaScript, PHP, Python, Ruby).

Cas concret

Nous allons montrer un exemple d'utilisation simple d'API : l'opération « Analyse Image » de l'API Computer Vision, à laquelle nous allons envoyer la photo ci-dessus : 2
La requête HTTP est la suivante :

```
POST https://westeurope.api.cognitive.microsoft.com/vision/v1.0/analyze?visual
Features=Tags,Faces,Description&language=en HTTP/1.1
Host: westeurope.api.cognitive.microsoft.com
Content-Type: application/json
Ocp-Apim-Subscription-Key: ••••••••••••••••••••••••••••••

{"url":"...url de la photo..."}
```

Le service renvoie une réponse au format JSON :

```
{
  "tags": [{
      "name": "monitor",
      "confidence": 0.99348324537277222
  }, {
      "name": "indoor",
      "confidence": 0.96023839712142944
  }, {
      "name": "screenshot",
      "confidence": 0.65517526865005493
  }, {
      "name": "flat",
      "confidence": 0.30609139800071716
  }],
  "description": {
      "tags": ["monitor", "indoor", "sitting", "computer", "top", "screenshot", "screen",
"microwave", "desk", "book", "table", "flat", "truck", "television", "laptop", "large",
"keyboard", "desktop", "cat", "bus", "white"],
      "captions": [{
```

```
        "text": "a screenshot of a computer screen",
        "confidence": 0.94612222280407776
      }]
  },…
  "faces": [{
      "age": 39,
      "gender": "Male",
      "faceRectangle": {
          "left": 982,
          "top": 531,
          "width": 50,
          "height": 50
      }
  }]
}
```


On peut constater que le service fournit un nuage de tags avec un score de confiance (notamment « écran » à 99%), et qu'il a également détecté un visage masculin. Certains tags de la description sont quant à eux plus incertains. Pour finir, l'API a donné un titre relativement correct.
Ces différentes informations peuvent ensuite permettre de procéder à un traitement : on peut imaginer des scénarios de détection d'objets, de texte, etc. Pour le traitement en particulier des visages, on préfèrera l'utilisation directe de l'API « Face ».

3

Améliorer mes résultats

L'amélioration continue est un levier très important de progression de certains services que vous exploitez. En effet, dès lors que ces services utilisent des données qui sont spécifiques à votre domaine d'activité / vocabulaire, il est évident que vous ne pourrez pas les alimenter dès l'origine d'une variété assez exhaustive de cas. Il faut donc les alimenter et corriger régulièrement, a minima pendant une période représentative de leur usage.

Parmi les exemples d'amélioration continue « automatique », on peut citer l'exemple de l'intégration du service « QnAMaker » dans le SDK du Bot Framework : lorsque le service hésite entre plusieurs questions correspondant à la demande d'un utilisateur du bot, il propose à cet utilisateur de choisir celle qui correspond en affichant une sélection multiple puis appelle une méthode spécifique « ActiveLearnAsync » pour remonter ce choix et ainsi s'enrichir. [Figure 3]

D'autres produits comme LUIS mettent des testeurs à disposition et permettent de récupérer les logs d'appels et faire de la complétion d'exemples a posteriori, mais de manière non automatique.

Combien cela coûte ?

Une fois la phase d'exploration gratuite passée, il est nécessaire de souscrire à des clés d'API plus importantes, dont la tarification est clairement accessible sur le portail Azure de Microsoft.
Etant donné qu'il s'agit d'APIs, il n'y a aucun coût d'infrastructure, de licence. Le coût se décompose généralement en une partie liée à la volumétrie d'appels (transactions) et une éventuelle liée au stockage d'informations spécifiques le cas échéant, comme pour les visages dans Face API.
A titre d'exemple de coûts :

- « LUIS » : 1,265 toutes les 1 000 transactions.
- « Computer Vision » : dépend des informations demandées : voir tableau ci-dessous.

La facturation des services cognitifs est visible directement sur le portail Azure [Figure 4].

NIVEAU	FONCTIONNALITÉS	TARIF
S1 - Jusqu'à 10 transactions par seconde pour les fonctionnalités suivantes :	Tag, Face, GetThumbnail, Color, and Image Type	0-1 million de transactions — 0,844 toutes les 1 000 transactions De 1 millions à 5 millions de transactions — 0,675 toutes les 1 000 transactions Plus de 5 millions de transactions — 0,549 toutes les 1 000 transactions
	OCR (avec impression), Contenu adulte, Célébrités et Élément géographique	0-1 million de transactions — 1,265 toutes les 1 000 transactions De 1 millions à 5 millions de transactions — 0,844 toutes les 1 000 transactions Plus de 5 millions de transactions — 0,549 toutes les 1 000 transactions
	Description et OCR (écriture manuscrite)	2,109 toutes les 1 000 transactions

NAME	TYPE	RESOURCE GROUP	COST (EUR)
NRO-Cognitive-FaceApi-WestEurop...	Cognitive Services	Cognitive_Demo	0.36
0f2d27dc-6004-4e1c-84c3-1b229...	Face Storage Included - Face API	--	0.00
ecff4b9d-5d93-456c-a0d0-1643...	Standard (in 1,000s) - Face API	--	0.36
NRO-Cognitive-ComputerVisionApi...	Cognitive Services	Cognitive_Demo	0.21
a68b7c67-c7da-4f84-9a5d-5256...	Standard S2 - Computer Vision API	--	0.01
c055322e-0b67-4024-92c1-6814...	Standard S1 - Computer Vision API	--	0.03
f20a5872-7aa2-432d-a3c7-5239...	Standard S3 - Computer Vision API	--	0.16
NRO-Cognitive-EmotionApi-WestUs...	Cognitive Services	Cognitive_Demo	0.01
dc5d40f7-474d-482a-b47e-ebb9...	Standard Emotion APIs (in 1,000s)	--	0.01

4

Et si on veut aller plus loin ?

A l'évidence, les services actuels sur étagère fournissent un panel important de solutions mais qui ne répondent pas forcément à toutes les contraintes. Ces derniers ont été créés pour répondre au plus grand nombre, entrainés avec des données classiques. Que se passe-t-il lorsque l'on utilise des cas aux limites ?

Reprenons l'exemple du chatbot et imaginons un utilisateur en milieu industriel, exposé à un niveau de bruit important, et devant utiliser sa voix pour transmettre sa demande : l'API Speech Bing que l'on utilise dans ce cas ne sera plus en mesure de retranscrire correctement le discours de l'utilisateur.
C'est ici que les services « personnalisés » viennent à notre secours : il est par exemple possible de fournir au service des exemples acoustiques spécifiques lui permettant d'améliorer le traitement ultérieur.

Enfin, si le large panel de services proposés sur étagère ou personnalisés ne répond pas encore aux problématiques, il reste la possibilité d'implémenter ses propres modèles en utilisant les outils de Deep Learning, en particulier la solution open-source Microsoft Cognitive Toolkit. Nous présenterons l'illustration de cet usage dans un prochain numéro.

Conclusion

Les services cognitifs ouvrent la voie à une démocratisation de l'intelligence artificielle dans notre quotidien et nos applications. Certains usages ont été rapidement identifiés mais beaucoup restent à inventer, et permettront sûrement d'enrichir les services existants et d'en développer de nouveaux.
Cette démocratisation est notamment permise par l'orientation « API », fournissant des services désormais interopérables et indépendants de la plateforme et du langage de programmation. Ils exposent le résultat d'un travail scientifique évolué, d'utilisation poussée du Machine Learning, en faisant abstraction de cette complexité.
Place maintenant à la créativité et aux nouveaux usages, tout en gardant en tête l'éthique. •

avanade

Raphael Bijiaoui
Responsable des offres digitales analytiques

Jonathan Petit
Manager et Data Scientist

Mettre en place un environnement de développement et d'expérimentation en Data Science

La data est le nouvel or noir pour les entreprises ; elles disposent souvent d'un trésor de guerre sans le savoir. Les initiatives IOT et Big Data décuplent encore leur capacité à accumuler de l'information.

Sous le terme d'intelligence artificielle se retrouvent de nombreuses activités : Acquisition de données, Exploration de données, Analyse Statistique, Machine Learning… Chacune de ces activités faisant appel à de nombreuses techniques aux noms exotiques pour les nouveaux venus dans le domaine : régression linéaire ou logistique, analyse en composante principale, séries chronologiques… Dans cet article, après un bref aperçu de ce qu'est la Data Science dans un premier temps, nous vous montrerons qu'il est simple de démarrer, grâce à une image de machine virtuelle sur étagère fournie par la plupart des opérateurs de Cloud. Dans notre cas, nous utiliserons Microsoft Azure et déploierons une machine virtuelle orientée Data Science : l'avantage de cette approche est que l'ensemble des outils principaux sont déjà préinstallés et préconfigurés. Vous avez le choix entre une version tournant sous Windows et une autre sous Linux. Les outils déployés dans les deux cas sont souvent identiques, même pour les produits Microsoft, ce dernier éditant désormais des solutions, de base de données notamment (SQLServer) sous Linux. A vous de choisir en fonction de vos connaissances, de vos préférences et de vos contraintes personnelles. Dans un troisième temps, nous réaliserons une première expérimentation avec Azure Machine Learning et aborderons la création, le test et le déploiement de notre expérimentation.
Enfin, nous ouvrirons la discussion pour vous indiquer quelques ressources sur internet qui vous permettront d'aller plus loin.

Data Science 101

L'objectif premier de la Data Science, et donc du Data Scientist qui la met en œuvre, est d'extraire de l'information des données qu'il a à disposition. Pour accomplir son objectif, le Data Scientist s'appuie sur un panel de connaissances et de techniques très variées, développées par des universitaires ou des ingénieurs, qui ont en commun de reposer sur des fondements mathématiques statistiques et de faire appel à une bonne dose d'expérimentation. L'essor de cette discipline et l'engouement qu'elle suscite sont fortement liés au contexte technologique actuel où l'IOT et le Big Data favorisent l'accumulation de données de toutes sortes : une mine d'or pour les Data Scientists et les entreprises pour lesquelles ils travaillent.

Définition :
Les travaux de data science ont pour objectif de faire ressortir des effets non facilement détectables d'un jeu de données. Ces effets entre les facteurs ne sont donc pas détectables « à l'œil nu » mais des analyses de données plus ou moins poussées aident à les identifier.
La data science se situe à l'interaction entre la capacité d'appréhender des problématiques scientifiques et des compétences de lecture, de transformation, de détection de ces phénomènes au cœur de la data. Le Data Scientist possède des compétences en modélisation et en interprétation de données de formats multiples. De plus en plus souvent viennent s'ajouter des compétences en infrastructure et architecture de données récentes.
De l'analyse de corrélations linéaires simples à la mise en place de réseaux de neurones censés imiter le fonctionnement du cerveau humain, il a à sa disposition toute une panoplie de méthodologies empruntées aux différentes disciplines scientifiques et interagissant avec elles à travers l'analyse de données.
De la mise en place de prévision de comportements sociologiques (séries chronologiques) à la compréhension de phénomènes physiques, son apport va grandissant dans tous les domaines de la recherche :
http://www.astronomy.com/news/2017/12/the-lsst-and-big-data-science

Data Science et Big Data
Du fait de ces volumes croissants de données, deux champs méthodologiques sont actuellement utilisés de plus en plus souvent conjointement par les Datas Scientists : les frameworks de calculs optimisés et les algorithmes d'approximation. Ces nouveaux frameworks de gestion de données distribuées intègrent dorénavant des capacités de calculs distribuées (auparavant il s'agissait uniquement de données distribuées). Les algorithmes d'approximation trouvent toute leur place au sein des traitements de données en environnement Big Data : il s'agit de modèles probabilistes (réseaux de neurones, algorithmes génétiques, …) beaucoup plus rapides que des calculs exhaustifs, maintenant toutefois un niveau de précision très élevé.

Le métier de Data-Scientist
Son rôle revient souvent à optimiser une problématique, une fonction : gagner du temps, maximiser le gain, minimiser l'erreur. Cela correspond souvent à la remise en question de processus existants. Ainsi, les données à valeur correspondent à l'analyse de l'historique des utilisations de ces processus, permettant d'identifier les points de rupture, les zones de bifurcations ou d'abandon du processus. Si l'on prend l'exemple de l'amélioration continue de la capacité d'un ChatBot à répondre correctement sur un certain nombre de sujets, le

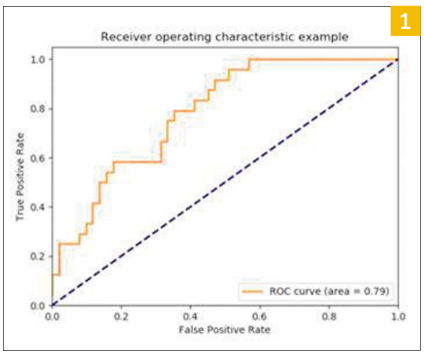
Data Scientist va construire son algorithme en ayant pour objectif de détecter les cas d'échecs et de les modifier de manière la plus automatisée possible.

Evaluation de la performance de projets de Data Science

Enfin, la performance des projets de Data Science se mesure souvent à l'aune d'un comportement purement aléatoire. Dit autrement, cela revient à estimer l'écart entre un résultat délivré par une méthodologie de Data Science et ce même résultat sans faire intervenir cette méthodologie. Les méthodes d'échantillonnages sont nécessaires à l'estimation de cette performance, la comparaison entre un échantillon "test/placebo" et le reste de la population permettra de construire ces indicateurs. C'est l'idée d'une base, d'un référentiel auquel se comparer, quitte à le construire dans cet unique objectif. La matrice de confusion, les courbes de Lift et de ROC sont également de bonnes représentations de la mesure de la performance de ces modèles heuristiques. 1

Restitution des travaux de Data Science :

Au-delà de l'analyse des résultats, un modèle de Data Science doit répondre au critère de pouvoir explicatif : cela indique que le modèle est maîtrisé et que les résultats qu'il fournit restent cohérents, particulièrement s'il s'agit d'un modèle auto-apprenant. Connecter un outil de monitoring aux étapes clés de transformation du modèle est souvent requis pour assurer sa robustesse et sa pérennité. La notion de "toutes choses égales par ailleurs" permet d'analyser les impacts de façon isolée et de conserver ce caractère intelligible.



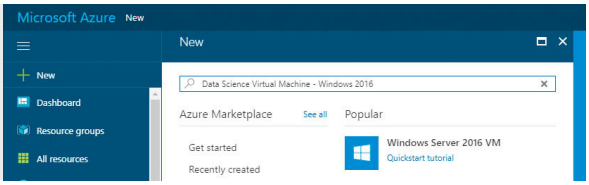
1

Crédit photo : https://fr.m.wikipedia.org/wiki/Courbe_ROC

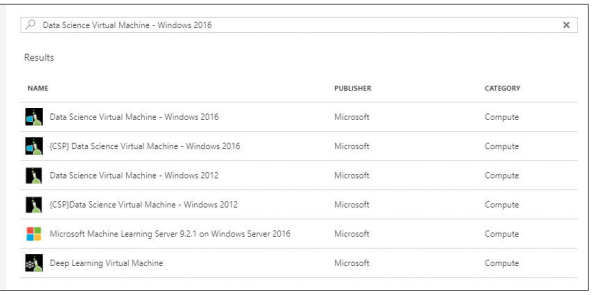
Création d'une virtual machine Azure Data Science

Pour démarrer la création de votre environnement de travail, connectez-vous au portail Azure (https://portal.azure.com/). Si vous ne disposez pas d'accès à Azure, vous pouvez créer un compte d'essai qui vous octroiera un budget mensuel à dépenser en services et vous permettra de tester la plupart des outils et services.

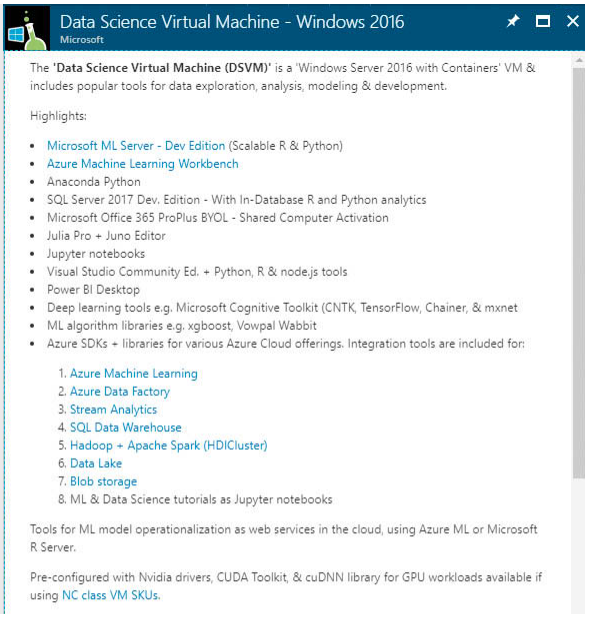
Après vous être connecté au portail Azure, cliquez sur New en haut de l'écran à gauche. Dans la zone de recherche, tapez : « Data Science Virtual Machine - Windows 2016 » puis « Entrée ».



Dans la liste de ressources retournées, sélectionnez l'image correspondante.
La liste présente les options sous Windows. N'hésitez pas à rechercher leur équivalent sous Linux si vous le souhaitez.



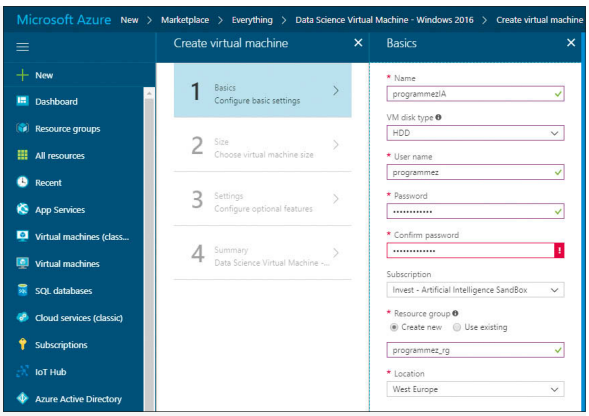
Un nouveau panel détaille alors le contenu de la machine qui va être créée. La configuration est très complète et contient l'essentiel des outils utiles à un Data Scientist. Il vous sera toujours possible par la suite d'ajouter de nouveaux outils si vous ne trouvez pas votre bonheur ici.



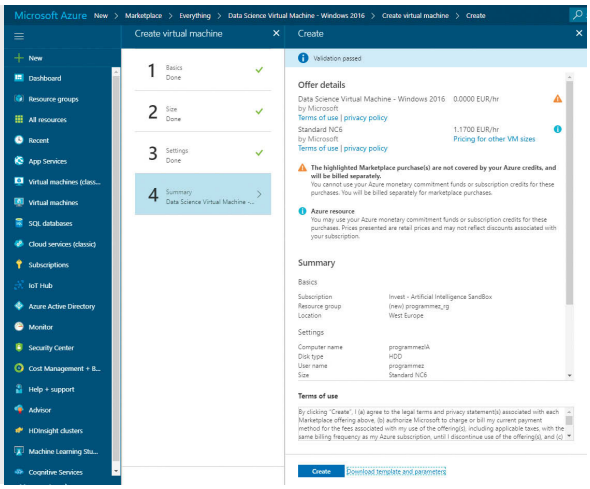
Il ne reste plus qu'à saisir les paramètres habituels nécessaires à la création d'une Virtual Machine dans Azure :

- le nom de la machine,
- le type de disque (SSD ou HDD),
- un user et mot de passe,
- la souscription utilisée, le nom d'un groupe de ressources et la localisation, soit le data center Microsoft dans lequel la machine sera déployée.

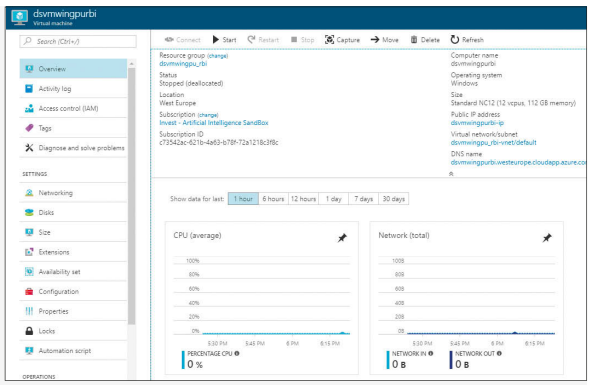
Une remarque concernant le type de disque : si vous souhaitez sélectionner une taille de machine (Etape 2 : Sizing) incluant un ou plusieurs GPUs, il est nécessaire à l'heure où cet article est écrit de sélectionner un disque HDD, le SSD n'étant pas supporté.



Passez sur le reste des options de taille et les Settings et validez la configuration à l'étape quatre. Si vous souhaitez réitérer la création d'une machine virtuelle avec des paramètres similaires, vous pouvez télécharger un script ARM à cette étape en cliquant sur le lien près du bouton « Create ».

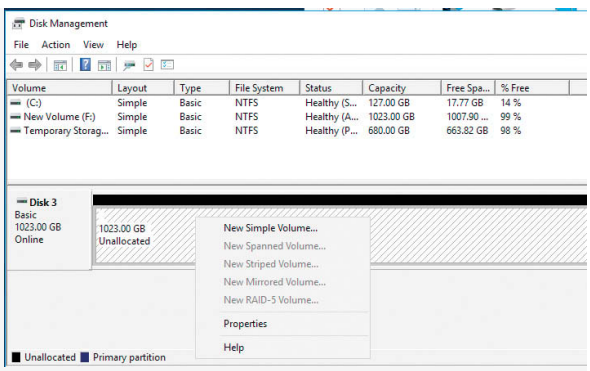


Une fois la machine créée, on peut s'y connecter à l'aide d'un fichier de connexion téléchargeable depuis l'écran de supervision de la machine dans le portail Azure. En cliquant sur « Connect », le fichier de connexion est généré automatiquement. Sauvegardez-le ou ouvrez-le.
Une remarque concernant les machines virtuelles Linux : pour s'y connecter il est nécessaire d'utiliser un outil tiers, non disponible par défaut dans Windows. Si vous n'en connaissez pas, vous pouvez utiliser X2GO par exemple.



Par défaut la machine virtuelle ne dispose pas d'une grande capacité de stockage mais seulement d'un disque système. Nous allons donc ajouter un disque supplémentaire afin d'étendre sa capacité. Ce sera l'endroit idéal pour stocker vos fichiers de données et le résultat de vos expérimentations.
Dans l'écran de supervision de la VM dans le portail Azure, sélectionnez les propriétés des disques dans le panneau de gauche. Dans le panneau de droite, cliquez sur « + Add data disk ». Vous pouvez attacher un disque existant ou un créer un nouveau. Choisissez « Create Disk » et remplissez les propriétés dont le nom du volume. Sauvegardez.

Une fois cela fait, connectez-vous à la Virtual Machine, il reste à rendre le disque disponible sous Windows. Ouvrez depuis le panneau de configuration le gestionnaire de disque. A la première ouverture, il reconnaît le disque. Acceptez sa proposition.
A ce stade, votre nouveau disque n'est pas encore un volume reconnu par Windows. Dans la liste des disques, repérez votre nouveau disque, et cliquez droit sur lui, sélectionnez « New simple volume … ». Dans le Wizard, allez jusqu'au bout des écrans et cliquez sur « Finish ». Après quelques instants, vous disposez d'un nouveau Volume et pouvez y accéder depuis l'explorateur de fichiers. Pour finir, lorsque vous accèderez au volume pour la première fois, Windows vous proposera de le formater.



Vous disposez désormais d'un environnement d'expérimentation opérationnel. N'oubliez pas d'éteindre votre machine virtuelle entre deux utilisations pour ne pas épuiser votre forfait trop rapidement. Nous vous conseillons d'activer les options d' « Auto Shutdown » depuis le tableau de bord de supervision de la machine virtuelle.

Exemple pratique

Algorithme de recommandation de lectures et de produits basé sur l'observation de comportements médias.

Dans cette partie, nous allons vous présenter la réalisation d'un projet de Data Science réalisé en partie en langage Python, en nous basant sur l'outil Azure ML inclus dans tout environnement Azure.

Commençons par une rapide introduction d'Azure ML :

Il s'agit d'une interface intégrée à Azure dédiée à la réalisation de travaux de Data Mining : modélisation, scoring, segmentations… La plupart des modèles de Data Science sont disponibles "sur étagères", il convient simplement de les paramétrer correctement. L'intérêt d'Azure Machine Learning est de pouvoir déployer un modèle en production en tant que service web en quelques minutes. Ce service web peut être appelé depuis n'importe quel appareil, quel que soit le lieu et utiliser toutes les sources de données à disposition. Une alternative est d'utiliser le WorkBench d'Azure ML. Cet outil complémentaire à Azure ML permet de faire évoluer vos « experiments » en mode desktop, tout en assurant une communication en continu avec Azure ML.
La documentation pour mettre en place un environnement Azure ML est très claire, le seul prérequis étant d'avoir un compte Azure : https://docs.microsoft.com/fr-fr/azure/machine-learning/studio/create-workspace
L'écran ci-contre représente un « experiment » de Azure ML. C'est cet « experiment » que nous transformerons en web-service lorsqu'il sera abouti. Un « experiment » attend en entrée un certain nombre d'informations sources auxquelles il appliquera les opérations nécessaires à l'exposition d'une sortie contenant les informations souhaitées. Les entrées en sorties doivent respecter les formats prédéfinis au sein de l' « experiment ». Un « experiment » est composé de blocs distincts et reliés entre eux. Chaque bloc réalise une partie des opérations de l' « experiment ». Les blocs sources contiennent les liens vers les données stockées dans votre environnement Azure, dans un élément de blob-storage. Chaque bloc, selon son rôle contient une ou plusieurs entrées et sorties. Les blocs aux fonctionnalités prédéfinies se trouvent sur la partie gauche de l'écran. La partie droite est consacrée aux détails correspondant aux blocs actifs. 2

Poursuivons cette partie par la description du projet implémenté :

Ce projet a pour finalité de proposer la consultation d'informations pertinentes en fonction des domaines d'intérêts des collaborateurs au sein d'une entreprise. Un projet unique peut comporter plusieurs « experiments ». L'objectif ici est de trouver les meilleures correspondances entre les goûts exprimés et les différentes bases de produits ou d'articles préalablement annotés. Ce calcul d'adéquation permet à l'algorithme de proposer des sujets ou des produits en lien avec les affinités déclarées ou observées (comportements de likes) des utilisateurs et les différentes sources connectées.

Le projet a été intégralement construit sur Azure Machine Learning pour profiter d'un certain nombre de fonctionnalités préconstruites, de faciliter le co-working à distance et l'interaction entre les développeurs et les Data Scientists.

Dans sa 1ère version, il couvre uniquement le sujet des voyages. Il pourra être élargi à d'autres thèmes affinitaires.

Le thème du voyage est en réalité découpé en une multitude de sous-catégories qui lui sont liées : plage (visuel), vacances, nature, pays, villes (géographique), …

Détaillons la liste des éléments nécessaires au fonctionnement du service web :

Les entrées (sources) :

- Liste de sources identifiées autour des sujets du voyage : blogs de voyageurs, produits liés au voyage, sites de réservations de vols, de trains (Source 1) ;
- Données historiques des réactions aux informations soumises sur ce thème par utilisateur (Source 2 : Positive, Negative, Etonnement).

La transformation des informations en entrée : nettoyage des données, remplacement des valeurs manquantes (moyennes par groupes homogènes). Il s'agit des blocs Python visibles sur l' « experiment » mais non détaillés ici. Ils correspondent aux opérations classiques de traductions de phrases en objets à des fins de comparaisons :

- Tokenizing (incl. N-Grams)
- Remove stop word
- Stemming

Ces applications diffèrent selon le langage, en anglais ici.

Plus loin sur le stemming :
https://github.com/snowballstem/pystemmer

L'algorithme de string-matching :
Distance de levenshtein entre les goûts des utilisateurs et les différentes sources.

Sortie :
Par utilisateur, le résultat obtenu est un score affecté aux différentes sources en entrée (source 1). Sélection des n sources aux scores les plus élevés.

En jaune, les exports de Data permettent d'itérer sur la formulation du score, en fournissant les calculs de distance détaillés.

En vert, la librairie JellyFish contient la fonction levenshtein, permettant une évaluation de la distance entre les phrases.

Publication du Service Web :
Une fois votre « experiment » terminé, c'est-à-dire que vous avez précisément formaté et défini les éléments attendus en entrée et en sortie, vous pourrez le transformer en Service Web en cliquant directement en bas de page « Deploy Web Service ». Vous disposerez alors d'une API Key pour utiliser ce service et éventuellement le connecter à d'autres services web.

Le flux entrant du service web enregistre en temps réel de nouveaux comportements utilisateurs (son activité) et vient moduler le score de celui-ci sur chacune des catégories en fonction de ces nouvelles informations. Il vient ainsi mettre à jour la base de score 2 en fonction des nouvelles observations.

Par exemple : une action de clic sur une photo de plage (annotation automatique), de hamac, de montagne ou de glace engendre un enregistrement et une interaction avec la base d'informations « voyages » constituée des 2 sources. Le score de cet utilisateur avec chaque produit est modulé suite à ce nouvel événement.

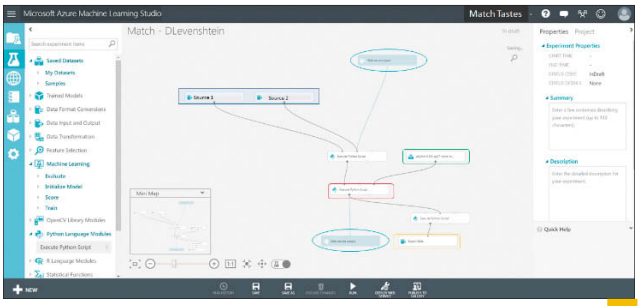
Démonstration de l'utilisation du service web :
Une nouvelle activité utilisateur est générée : « Picture, Beach, Hot&Clear, Chili, Like ».

« Image, Beach, Chili » correspondent à des informations de type source 1 : [format : image, ambiance : plage, climat : top, géographique : Amérique du Sud]

« Waouh » correspond à une information de type source 2 : type de réaction.

Le type de réaction « Waouh » va permettre d'utiliser les termes de type source 1 avec l'ensemble des informations préexistantes dans la source 1. Pour cet utilisateur, l'ensemble de ces informations qui matcheront le plus entre elles (selon la distance de Levenshtein ici) avec cette nouvelle activité verront accroître leurs scores respectifs.

Ainsi, les propositions de contenus sont



2

optimisées par rapport à une proposition aléatoire de ces mêmes contenus. Mais attention, conseil : le fait de préserver une partie d'aléatoire dans ce type de sujets vous aidera à détecter des interactions imprévisibles par ce type de méthodologie.

Voici le script que nous avons utilisé : retrouvez le code en ligne sur GitHub :
https://github.com/Jostatds/Match_Levenstein/tree/master

Tests :
La qualité du score de match a pu être améliorée dans un premier temps grâce à un export des résultats et un travail de tâtonnement. Une méthodologie de comptage des fréquences d'apparitions relatives de types de contenus peut être implémentée. C'est en quelque sorte une gestion du caractère de rareté de certaines occurrences. L'observation des usages permet la détection de corrélations entre mots-clés et sujets. Enfin, la mesure du taux de succès se fait via l'observation des comportements de consommations des contenus suggérés (temps passé sur une page, transformation).

Annotation de contenus :
Si besoin, les images peuvent être automatiquement annotées en utilisant l'API Microsoft de Computer Vision :
https://docs.microsoft.com/en-us/azure/cognitive-services/computer-vision/home

A partir de ces annotations, un premier service web peut être utilisé pour associer les termes les plus significatifs (probabilités seuils) à un document donné (image, texte, site web…).

Pour aller plus loin

Vous disposez désormais de votre environnement et avez survolé une première expérimentation. Pour aller plus loin, Microsoft propose en partenariat avec le site de MOOC en ligne EDX un parcours de certification sur la Data Science (https://academy.microsoft.com/en-us/professional-program/).

Si vous êtes intéressé pour suivre les cours, ils sont disponibles gratuitement. Seule la certification est payante. •

Christophe Villeneuve
Consultant IT pour Ausy, Mozilla Rep, auteur du livre "Drupal avancé" aux éditions Eyrolles et auteur aux Editions ENI, PHPère des elePHPants PHP, membre des Teams DrupalFR, AFUP, LeMug.fr (MySQL/MariaDB User Group FR), Drupagora...

Les WebExtensions

PARTIE 1 : LA THÉORIE

Les modules complémentaires, ou extensions, révolutionnent le web en modifiant le comportement des navigateurs. Ainsi, aujourd'hui pour harmoniser le développement des applications, ce format a été pensé pour être compatible avec l'API d'extensions supportée par Firefox, Google Chrome, Opera, Edge, et les navigateurs compatibles QTWebKit, au prix de quelques modifications mineures.

Depuis de nombreuses années, le navigateur est la porte d'entrée vers Internet, car sans lui, vous ne pourrez pas accéder à l'information, aux contenus et aux nombreuses applications métiers (API) qui ne font que d'augmenter. Il a la lourde charge de transporter et d'afficher ces informations et d'améliorer leurs fonctionnalités avec des modules complémentaires.

Une WebExtension est le nom de l'API utilisée pour écrire des extensions pour Firefox. La technologie standard utilisée est HTML / CSS / JS, développée pour obtenir une compatibilité entre les navigateurs : dans une large mesure, l'API est compatible avec l'API d'extension supportée par Google Chrome, Opera, Edge, ce qui permet de créer et d'utiliser plus facilement les extensions sur les plateformes de navigateurs.

Par ailleurs, chaque navigateur propose des fonctionnalités, options, et des APIs supplémentaires. C'est pourquoi nous nous appuierons sur les WebExtensions proposées par la fondation Mozilla, disponibles à partir de Firefox 'Quantum' 57 depuis le 14 novembre 2017 pour être ISO fonctionnelles avec tous les autres.

Enfin, les WebExtensions disposent de plusieurs options d'interface utilisateur afin que leurs fonctionnalités puissent être mises à la disposition de l'utilisateur. Elles sont déclarées et appelées à partir d'un fichier manifest.json. 1

Manifest

Le fichier manifest.json est le seul fichier obligatoire dans une webExtension. Il est formaté au format JSON sur le principe de clés/valeurs et servira de point d'entrée à votre application.

Ce fichier est important, car vous spécifiez un certain nombre de métadonnées obligatoires comme le nom, le numéro de version, la version du manifest et se présente de la manière suivante :

```
{
 "manifest_version": 2,
 "name": "Programmez",
 "version": "2.0"
}
```

pour obtenir le résultat de l'image suivante : 2

Il y a de nombreuses informations complémentaires et indispensables comme la description et l'affichage d'une icône pour identifier facilement sa webExtension dans la page des modules complémentaires.

```
 "description": "description de la WebExtension",
 "icons": {
  "19": "icons/icon-19.png",
  "48": "icons/icon-48.png"
 },
```

Permissions

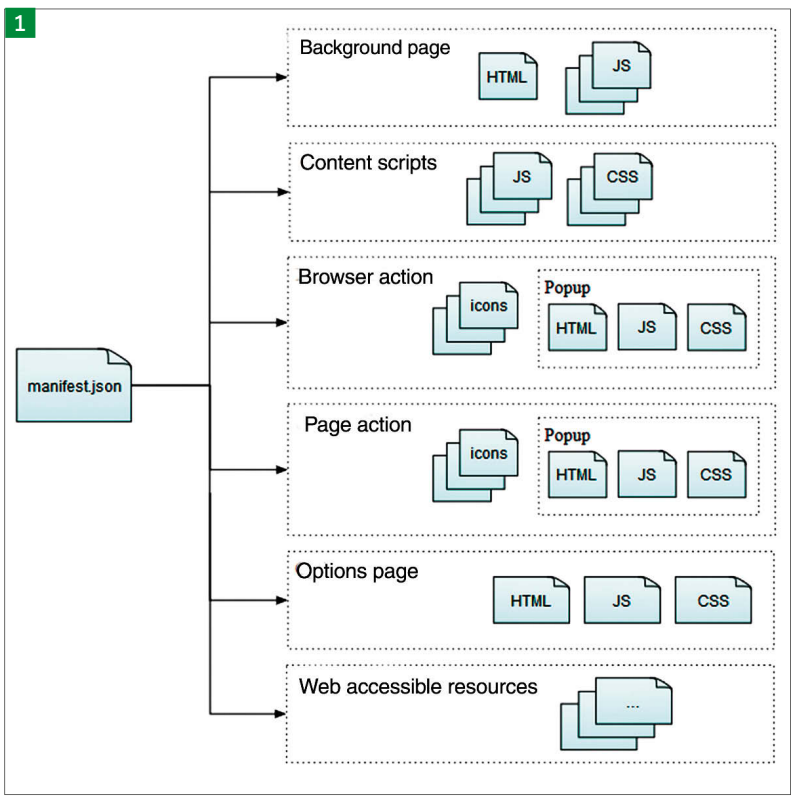
Les WebExtensions permettent aux extensions de demander des permissions supplémentaires lors de l'installation. Ainsi pour accéder aux nombreuses APIs à partir des WebExtensions, il est nécessaire que l'utilisateur accorde les accès dont elle a besoin pour utiliser pleinement l'application. 3

L'utilisation de la clé 'permissions' permet d'obtenir des accès supplémentaires à votre extension, dont le navigateur informe l'utilisateur au moment de l'installation.

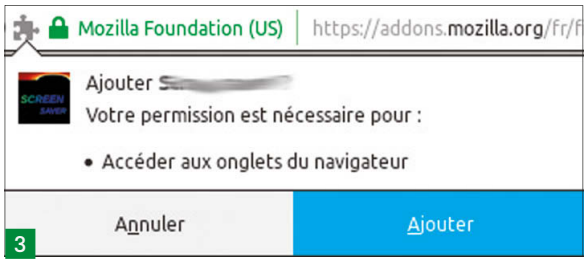
La clé peut contenir trois types d'autorisations (non obligatoire) :

• Les permissions d'hôte ;

1



2



• Les permissions API ;
• La permission activeTab.
Elle se présente de la manière suivante :

```
"permissions": [
 "*://developer.mozilla.org/*",
 "windows",
 "activeTab",
]
```

L'argument 1 correspond aux permissions d'hôte qui correspond aux modèles de correspondance, comme une URL.
L'argument 2 correspond aux permissions API, c'est-à-dire les mots clés et chaque mot clé comme une API WebExtension que l'extension souhaite utiliser comme 'windows' pour interagir avec les fenêtres.
L'argument 3 'activeTab' correspond à l'onglet actif, permettant d'interagir avec l'extension et l'utilisateur.
Documentation MDN :
https://developer.mozilla.org/fr/Add-ons/WebExtensions/manifest.json/permissions

Arrière-plan

La clé d'arrière-plan s'appelle 'background' et permet d'inclure un ou plusieurs scripts d'arrière-plan et éventuellement une page d'arrière-plan dans votre extension. Les scripts d'arrière-plan sont chargés dès que l'extension est installée et reste chargée jusqu'à ce que celle-ci, soit désinstallée ou désactivée. Pour cela, vous déclarez la clé 'background' dans votre fichier manifest.json qui appellera le fichier JavaScript de la façon suivante :

```
"background": {
 "scripts": [
  "background/loader.js"
  ]
}
```

Le chargement de plusieurs scripts en arrière-plan s'effectue de la manière suivante :

```
"background": {
  "scripts": ["jquery.js", "my-background.js"]
 }
```

Documentation MDN
https://developer.mozilla.org/fr/Add-ons/WebExtensions/manifest.json/arriere-plan

Les scripts de contenu

Les scripts de contenu sont déclarés avec comme clé de permissions 'content_scripts' dans le fichier manifest.json. Ils sont chargés en même temps que les pages web dont l'URL correspond à un modèle donné. Cette clé est un tableau comprenant des objets tels que :
• Matches : spécifie les modèles d'URL à associer afin que les scripts soient chargés ;
• JS : contient les fichiers Javascripts qui seront appelés.
Elle se représente de la manière suivante :

```
"content_scripts": [{
 "matches": [
  "*://mozilla.com/*",
  "*://localhost/*"
  ],
  "js": [
  "js/demo.js"
  ]
}],
```

Ici le script demo.js sera exécuté seulement pour les URL contenant Mozilla et Localhost.
Il est possible d'exclure certaines pages sans effectuer le reste des appels avec l'objet 'Exclude_matches' dans ce tableau et se représente de la façon suivante :

```
"content_scripts": [{
 "matches": [
  "<all_urls>"
  ],
  "exclude_matches": [
   "*://www.mozilla.org/*",
   "*://*.votreURL.org/*"
  ],
  "js": [
  "background/all-sites.js"
 ]
}],
```

Ici, le script JavaScript est appelé pour toutes les pages web sauf pour les sites Mozilla et localhost.
Documentation MDN :
https://developer.mozilla.org/fr/Add-ons/WebExtensions/manifest.json/content_scripts

Action de navigateur

Une action de navigateur est un bouton que votre extension ajoute à la barre d'outils du navigateur. Elle est déclarée avec comme clé de permission 'browser_action' dans le fichier manifest.json. Le bouton comporte une icône et peut éventuellement avoir une fenêtre contextuelle dont le contenu est spécifié à l'aide de HTML, CSS et JavaScript et se présente de la manière suivante :

```
"browser_action": {
 "default_icon": {
  "16": "button/icon-16.png",
  "32": "button/icon-32.png"
 },
 "default_title": "Titre",
 "default_popup": "popup/popup.html"
}
```

L'exemple montre la prise en charge de différents arguments comme :

- L'icône de l'action du navigateur doit pouvoir afficher une icône de différentes tailles suivant le navigateur ;
- Un titre qui s'affiche quand la souris passe par-dessus ;
- Une action à travers un popup.

Documentation MDN :
https://developer.mozilla.org/fr/Add-ons/WebExtensions/manifest.json/browser_action

Action de page

Une action de page est une icône que votre extension ajoute dans la barre d'URL du navigateur. Elle est déclarée avec comme clé de permission 'page_action' dans le fichier manifest.json. Les actions de page sont comme les actions de navigateur (voir plus haut), mais elles sont associées à des pages web particulières et non au navigateur dans son ensemble.
Le but de cette action est d'afficher une action pertinente sur certaines pages et non en permanence, ce qui est différent des actions de navigateur. Par défaut le bouton est caché dans tous les onglets et se présente de la manière suivante :

```
"page_action": {
  "default_icon": {
    "16": "button/icon-16.png",
    "32": "button/icon-32.png"
  },
  "default_title": "Titre",
  "default_popup": "popup/popup.html"
}
```

La description du code exemple est identique aux actions du navigateur sauf pour l'élément déclencheur.
Documentation MDN :
https://developer.mozilla.org/fr/Add-ons/WebExtensions/manifest.json/page_action

Page d'options

Une page d'options est une page de préférences ou d'options pour votre WebExtensions. Elle permet d'offrir des paramètres que les utilisateurs peuvent modifier. L'utilisateur peut accéder à cette page depuis le gestionnaire des extensions du navigateur et enregistrera les modifications lors de la sauvegarde.
Elle est déclarée par une clé de permission 'option_page' dans le fichier manifest.json et se présente de la façon suivante :

```
"options_ui": {
    "page": "options.html"
},
```

L'exemple montre que la page est un fichier HTML, et pour y accéder vous passerez par l'interface des modules complémentaires.

Documentation MDN :
https://developer.mozilla.org/fr/Add-ons/WebExtensions/manifest.json/options_ui

Les Ressources

Les ressources dans les webExtensions permettent d'associer et de lister toutes les images, scripts des pages ou de contenus pour les mettre à la disposition des pages web.
L'autre intérêt de stocker toutes ces données, est de vous éviter de faire appel à des serveurs distants, ce qui est utile si vous vous trouvez dans un lieu sans accès au web.
Elle est déclarée par une clé de permission 'web_accessible_resources' dans le fichier manifest.json et se présente de la façon suivante :

```
"web_accessible_resources": [
  "ressources/logo.png"
]
```

Documentation MDN :
https://developer.mozilla.org/fr/Add-ons/WebExtensions/manifest.json/web_accessible_resources

INTERFACES UTILISATEUR

Les interfaces ne se limitent pas au fichier 'manifest' décrit ci-dessus, car de nombreuses autres options visuelles sont utiles pour les utilisateurs -qu'il ne faut pas oublier - car suivant les navigateurs, certaines ont été rendues compatibles, mais elles sont intéressantes, car vous pouvez les utiliser dans votre navigateur Firefox.

Élément du menu contextuel

L'option d'interface utilisateur ajoute un ou plusieurs éléments dans un menu contextuel du navigateur, c'est-à-dire lorsque vous effectuez un clic droit avec votre souris, un menu appelé menu contextuel s'affiche. Il propose de nombreuses options pour exposer des fonctionnalités pertinentes de votre webExtension.
Pour utiliser cette fonction, vous devez déclarer un fichier JavaScript qui sera appelé avec la clé 'background' dans votre fichier manifest.json, décrite plus haut.

Documentation MDN :
https://developer.mozilla.org/fr/Add-ons/WebExtensions/user_interface/elements_menu_contextuel

Barre latérale

Une barre latérale est un volet qui s'affiche à gauche de la fenêtre du navigateur, à côté de la page web. Elle est déclarée par une clé de permission 'sidebar_action' dans le fichier manifest.json et se présente de la façon suivante :

```
"sidebar_action": {
  "default_icon": {
    "16": "button/icon-16.png",
    "32": "button/icon-32.png"
  },
  "default_title": "Ma barre latérale !",
  "default_panel": "sidebar/sidebar.html"
}
```

Lors de l'installation de l'extension, la barre latérale s'ouvre lorsque celle-ci est installée, puis obéit à la sélection de visibilité de la barre

latérale de l'utilisateur. L'interaction de l'utilisateur dans la barre latérale est traitée par son document HTML.

Documentation MDN :
https://developer.mozilla.org/fr/Add-ons/WebExtensions/user_interface/barres_laterales

Notifications

Les notifications vous permettent d'afficher des informations sur votre extension ou son contenu en utilisant le système d'exploitation sous-jacent.
Elle est déclarée par une clé de permission 'notifications' dans le fichier manifest.json et se présente de la façon suivante :

```
"permissions":
[
 "notifications"
]
```

Votre WebExtension déclenche un événement vers l'extension lorsque l'utilisateur clique sur une notification ou lorsqu'une notification se ferme (automatiquement ou à la demande de l'utilisateur).

Documentation MDN :
https://developer.mozilla.org/fr/Add-ons/WebExtensions/API/notifications

Conclusion

L'approche modulaire, en combinant HTML5 / CSS 3 et JavaScript, ouvre de nouvelles portes avec les nombreuses APIs disponibles et surtout le portage du web dans des secteurs qui ne sont pas attendus. Néanmoins, le fichier 'Manifest.json', continue à évoluer pour répondre aux nombreuses attentes des internautes dont nous aurons l'occasion de reparler. •

Liens utiles pour le fichier manifest dans :

- Firefox : https://developer.mozilla.org/fr/Add-ons/WebExtensions/manifest.json
- Chrome : https://developer.chrome.com/apps/manifest
- Edge : https://docs.microsoft.com/en-us/microsoft-edge/extensions/api-support/supported-manifest-keys

Démarrer avec les WebExtensions

- MDN web Docs : https://developer.mozilla.org/fr/Add-ons/WebExtensions/

Les WebExtensions

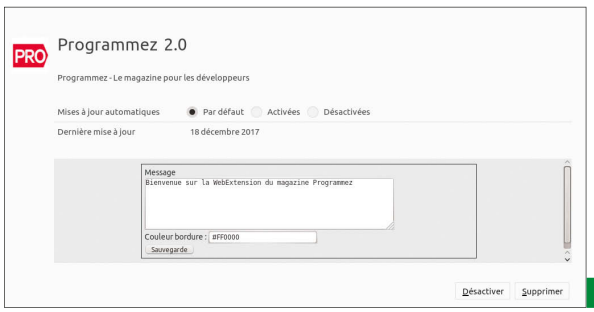
PARTIE 2 : LA PRATIQUE

Une extension est une collection structurée de code de développement web qui modifie le comportement ou l'apparence des navigateurs par exemple les extensions font partie de Firefox depuis de nombreuses années et permettent aux utilisateurs de personnaliser leur expérience de navigation.
A partir l'article précédent, nous vous proposons de construire votre propre WebExtension et de rentrer plus en profondeur dans le sujet. Le cas pratique que nous vous proposons, permet de personnaliser et d'accéder différemment à votre magazine préféré.
L'arborescence de notre webExtension se décompose en une architecture bien définie pour être le plus clair possible. 1
Nous ferons abstraction du fichier manifest.json, décrit dans l'article précédent. Cependant, nous vous mettons la visualisation globale possible de la structure du module. 2
L'extension proposée dans l'article montre plusieurs concepts élémentaires de l'API des WebExtensions comme :

- Ajout d'un bouton à la barre d'outil ;
- Ajout d'un lien au menu contextuel ;
- Définition d'une popup à l'aide de HTML, CSS et JavaScript ;
- Injection des content scripts dans chaque page web ;
- Communication entre les content scripts et le reste du module ;
- Mettre en package les ressources dans le module qui peuvent ensuite être utilisées par la webExtension.

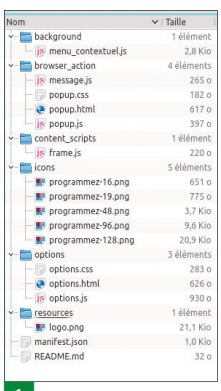
Les préférences

Les préférences sont les pages d'options; elles définissent les préférences de notre webExtension, que vous pouvez modifier. Vous accéderez à la page d'options du gestionnaire des modules complémentaires du navigateur ou à partir de la barre URL about:addons pour retrouver la ligne suivante : 3
Pour afficher cela, nous déclarons dans le fichier manifest.json, la clé 'options_ui' pour appeler notre fichier HTML. 4
Nous retrouvons les informations disponibles comme le logo, le

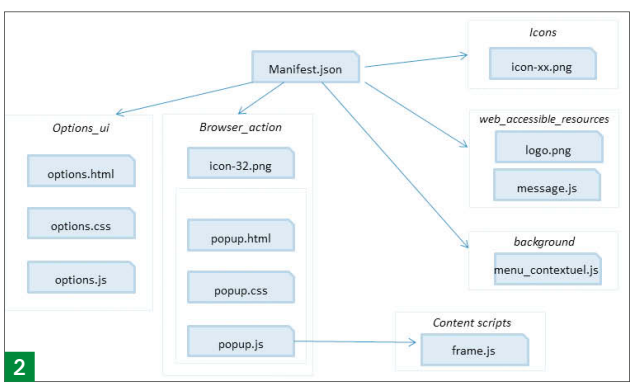


4

3



1



2

nom et le descriptif de la webExtension. Ensuite, vous trouverez l'écran de configuration avec 2 champs de saisie :

- Un champ 'message' qui sera affiché dans le menu de votre Extension ;
- Un champ 'couleur bordure' permettant de changer la couleur de contour des pages du site de votre magazine.

Le bouton sauvegarde enregistrera les changements.
Par ailleurs, le formulaire chargera une feuille CSS pour personnaliser l'écran et un fichier JavaScript sera appelé pour rendre dynamique la page.

Permissions

Pour utiliser cette option de stockage asynchrone, nous ajoutons la permission 'storage'

```
"permissions": ["storage"].
```

Chargement et sauvegarde

Le chargement et la sauvegarde des préférences passent par le même appel de la manière suivante :

```
var gettingItem = browser.storage.local.get(['settings']);
gettingItem.then(restoreSettings);
document.querySelector("form").addEventListener("submit", saveSettings);
```

Chargement

Le script charge les préférences de la webExtension, les affiches et les sauvegardera quand le bouton de Sauvegarde sera cliqué dessus. Lors du chargement de la page pour la première fois, le formulaire initialisera celui-ci avec les paramètres par défaut et rechargera le formulaire :

```
browser.storage.local.set({"settings" : settings});
window.location.reload();
```

Lors des prochains accès à cette page, les valeurs stockées seront envoyées au formulaire de la façon suivante :

```
document.getElementById("message").value = item.settings['message'];
document.getElementById("bordure").value = item.settings['bordure'];
```

Sauvegarde

Après les modifications du formulaire, vous sauvegardez les données qui se composeront de la manière suivante :

```
function saveSettings(e)
{
  settings['message'] = document.querySelector("#message").value;
  settings['bordure'] = document.querySelector("#bordure").value;

  browser.storage.local.set({"settings" : settings});

  e.preventDefault();
}
```

Ces données deviendront les valeurs par défaut à chaque utilisation de la webExtension et lors du prochain Affichage des préférences.

Écran Menu

L'écran menu est une action du navigateur représentée par un bouton, ajoutée à la barre d'outils de celui-ci. Les utilisateurs peuvent

5

cliquer sur le bouton pour interagir avec notre extension. Ce bouton restera visible tout le temps, quelle que soit votre page web affichée. Pour cela, la déclaration de notre application et l'affiche de l'icône seront déclarées dans le fichier manifest.json avec la clé 'browser_action' pour appeler notre page HTML. 5

Permissions

Pour utiliser cette option de stockage asynchrone, nous ajoutons la permission 'storage' et 'onglets'

```
"permissions": ["storage","tabs"].
```

Les actions du navigateur

Lorsque vous cliquez sur la webExtension, vous obtenez un formulaire contenant différentes informations dynamiques.

Logo

Le logo Programmez est chargé directement du dossier 'ressources'. Il a été déclaré dans le fichier manifest.json avec la clé 'web_accessible_resources'. Comme ceci, l'image affichée est chargée depuis l'extension et non venant d'un site distant, pour éviter les problèmes de connexion et de temps de chargement.

Message

Le message de bienvenue provient des préférences (voir partie plus haut) que nous utiliserons dans un fichier JavaScript, appelé 'message.js'

```
function onError(error) {
  console.log(`Error: ${error}`);
}

function onGot(item) {

  document.getElementById("message").textContent = item.settings['message'];

}
browser.storage.local.get('settings').then(onGot, onError);
```

Il va recharger le message à chaque ouverture de la fenêtre contextuelle à partir de la barre d'outils.

Les Boutons

Les boutons affichés, sont des actions qui ouvrent un nouvel onglet vers une nouvelle URL. Ici, nous accéderons à la page d'accueil de la page Programmez et l'autre l'accès à votre compte.

```
(function (browser) {
  "use strict";

  document.getElementById("programmez").onclick = function () {
    browser.tabs.create({
      "url": "https://www.programmez.com"
    });
  };
```

```
document.getElementById("votreCompte").onclick = function () {
  browser.tabs.create({
    "url": "https://www.programmez.com/user"
  });
};
})(chrome || browser);
```

Le script montre l'élément ID du bouton; nous utiliserons API tabs pour créer une nouvelle page et saisir l'URL définie.

Menu contextuel

À partir du clic droit de la souris, nous afficherons le menu contextuel avec un élément venant de notre webExtension. 6

Permission

Pour utiliser cette fonctionnalité, nous demandons la permission 'contextMenus' dans votre manuifest.json pour utiliser l'API, comme ceci :

```
"permissions": [
  "contextMenus"
]
```

Actions

Dans le menu contextuel, nous ajoutons une nouvelle ligne comprenant le logo et le titre de notre webExtension, fourni par le fichier manifest. Comme il s'agit d'un lien permanent, nous le rendons générique pour faciliter la disponibilité dans tous les onglets (background) du navigateur. Nous ajoutons le point d'entrée suivant dans notre fichier manifest.json :

```
"background":
{
  "scripts": [
    "background/menu_contextuel.js"
  ]
},
```

Lors du chargement de la webExtension, nous créons une ligne dynamique avec un lien cliquable.

```
function genericOnClick(info, tab) {
 if (info.menuItemId == "siteInternet")
 {
  browser.tabs.create({
      "url": "https://www.programmez.com"
    });
 }
}

var menuContextuel = browser.contextMenus.create({
 "id":"siteInternet",
 "title": "Programmez",
 "onclick": genericOnClick
});
```

Lorsque l'utilisateur clique sur ce menu, nous générons une action 'genericOnClick', qui vérifiera la provenance pour ensuite créer un nouvel onglet vers le site de votre magazine.

Scripts de contenu

Le script de contenu que nous vous proposons permet d'habiller les pages web du site Programmez, en ajoutant une bordure de couleur de votre choix définie par les préférences. La valeur proposée par défaut est le rouge, dont la valeur est Red ; en hexadécimal, le code valeur sera #FF0000.

Déclaration

Nous déclarons dans notre fichier manifest.json, le script JavaScript appelé, associé au site 'programmez'.

```
"content_scripts": [
 {
   "matches": ["*://*.programmez.com/*"],
   "js": ["content_scripts/frame.js"]
 }
],
```

Exécution

Le fichier 'frame.js' sera appelé à chaque action aussi bien à partir du menu contextuel que du bouton de la barre d'outils. 7

```
function onError(error) {
  console.log(`Error: ${error}`);
}

function onGot(item) {
 document.body.style.border = "20px solid " + item.settings['bordure'];
}

browser.storage.local.get('settings').then(onGot, onError);
```

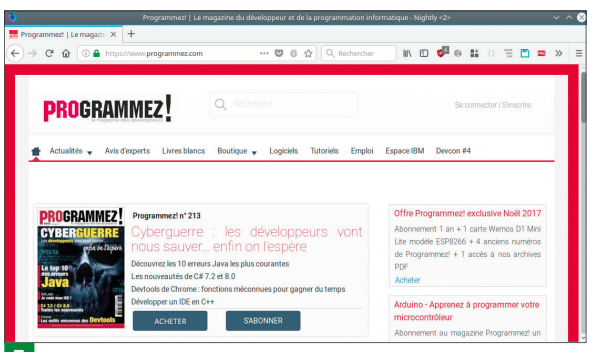
A chaque chargement de la page du site Programmez, le script récupère les paramètres en mémoire pour lui ajouter une bordure de couleur.

Source et déploiement

Le code source de la webExtension est disponible sur le site internet du magazine et compatible pour l'ensemble des navigateurs. Pour l'installer dans votre navigateur Firefox, il vous suffit de taper l'URL about:debugging et de charger le fichier XPI.

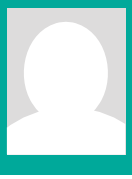
Conclusion

Comme vous pouvez le voir, le cas pratique nous a présenté la communication entre quelques applications et les accès aux permissions. Nous vous conseillons de vous rendre sur la documentation pour en découvrir plus. •



7

6



Philippe Lorieul,
Cellenza

cellenza
DOESITBETTER | Conseil - Expertise Microsoft & méthodes agiles

10 erreurs qui facilitent la vie des hackers

En ingénierie logicielle, on parle souvent de bonnes pratiques. Les sujets tels que les règles de nommage, les design patterns, les principes SOLID et le clean code sont traités en profondeur dans la littérature et sur internet. On trouve en revanche beaucoup moins d'information sur les pièges à éviter et les anti-patterns. Pourtant apprendre à reconnaître ces pièges est bien souvent la méthode la plus efficace pour ne pas tomber dedans. Dans cet article, nous allons détailler les 10 principales erreurs de programmation qui facilitent la vie des hackers.

1 Ne pas valider les entrées

C'est probablement l'erreur la plus connue. Et pour cause, elle est à l'origine d'un grand nombre de vulnérabilités, tous types d'applications confondus (web, serveur, …).

Pour être utile, une application doit pouvoir échanger avec le monde extérieur. Ainsi, elle va recevoir des données de différentes sources : d'utilisateurs, de services métiers, de bases de données, de fichiers ou de tout autre système informatique.

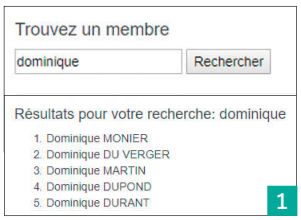
Ces acteurs externes ne sont pas sous notre contrôle et rien ne les empêche de fournir à notre programme des données invalides ou nuisibles. Charge au développeur de contrôler ces entrées, faute de quoi son application risque d'être compromise. Une des conséquences d'un défaut de validation est la faille appelée cross-site scripting ou XSS, qui permet à un attaquant d'exécuter du code dans le navigateur des utilisateurs d'un site vulnérable.

Imaginons une application web telle qu'un réseau social contenant un champ de recherche permettant de trouver un membre du site. Lorsqu'un utilisateur entre un nom et soumet sa requête, la saisie est rappelée sur l'écran de résultats. 1

```
<h2>Trouvez un membre</h2>
<form action="/search/results" method="get">
  <input name="query" id="query" type="text" value="" />
  <button id=="submit" type="submit">Rechercher</button>
</form>
```

Tant que la saisie reste dans le cadre de l'utilisation attendue de la fonction de recherche, tout va bien. Si en revanche on entre la recherche suivante :

```
<script language="javascript">alert('xss');</script>
```



Le texte de la recherche va être inclus dans le code source de la page, et le code Javascript inséré est exécuté par le navigateur.

Code de la page de résultats :

```
<h2>
  Résultats pour votre recherche:
  <script language="javascript">alert('xss');</script>
</h2>
```
2

Pour effectuer une recherche par nom et prénom, l'utilisateur a uniquement besoin des lettres de l'alphabet et éventuellement de quelques caractères supplémentaires, comme l'apostrophe ou le trait d'union. En acceptant seulement les caractères utiles dans notre programme, nous empêchons la saisie de la balise `script` et réduisons le risque d'introduire une faille XSS.

Le cross-site scripting n'est pas le seul risque introduit par un défaut de validation. D'autres injections (LDAP, SQL, commande), les débordements de tampon, les dépassements d'entiers et bien d'autres encore sont susceptibles d'envahir notre code si l'on ne prend pas soin de valider correctement les données entrant dans notre système.

2 Ne pas assainir les données envoyées

L'échange de données entre notre application et les acteurs externes est rarement unidirectionnel. La plupart du temps, nos programmes envoient eux-mêmes des données à d'autres systèmes.

Ces systèmes n'ont pas connaissance du contexte qui est à l'origine des données qui leurs sont transmises. Dans l'exemple du XSS présenté avant, un navigateur ne fait pas la différence entre le code HTML légitime de votre application et celui saisi par un attaquant. C'est donc à notre logiciel de préciser la nature de l'information qu'il fournit. Dans le code ci-dessous, la ligne 3 n'a pas été insérée par le développeur. Elle ne devrait donc pas être interprétée comme telle par le navigateur.

```
1 <h2>
2   Résultats pour votre recherche:
```

```
3     <script language="javascript">alert('xss');</script>
4  </h2>
```

Pour faire la distinction entre le code « interprétable » et la donnée saisie par l'utilisateur, certains caractères doivent être encodés. Le code source de la page de résultat devrait ressembler à celui-ci :

```
1  <h2>
2     Résultats pour votre recherche:
3     &lt;script language=&quot;javascript&quot;&gt;alert(&#39;xss&#39;);&lt;/script&gt;
4  </h2>
```

Dans cette version, les caractères réservés en HTML (chevrons, quotes simples et doubles), ont été remplacés par des entités (> et <). Ces combinaisons de caractères servent à indiquer au navigateur la différence entre code à interpréter et données à afficher à l'écran. Le résultat est visible dans la page. 3

L'outil à utiliser pour neutraliser les attaques de ce type dépendent des technologies choisies pour les développements. En php par exemple, c'est la fonction `htmlspecialchars()` qu'on appellera dans la page de résultats. Certains frameworks comme ASP.NET MVC prennent une position sûre et nettoient les données par défaut. Il faut invoquer explicitement une méthode risquée, `Html.Raw()`, pour qu'une faille XSS puisse apparaître.

Là encore, le cross-site scripting ne sert que d'exemple. D'autres vulnérabilités sont corrigées en assainissant les données envoyées aux systèmes extérieurs. Les requêtes paramétrées en SQL permettent d'éviter les injections SQL, l'encodage de caractères protège des injections LDAP et des XSS. Le principe est toujours le même : il faut assainir les données que l'on envoie aux autres systèmes.

3 Laisser les erreurs techniques remonter jusqu'à l'utilisateur

Les messages d'erreur générés par les applications constituent une mine d'informations utiles aux développeurs. Seulement, s'ils peuvent aider un programmeur à trouver un bug dans son application, ces messages peuvent aussi aider un attaquant y trouver une faille. Les erreurs mal gérées dans notre code peuvent être exposées à nos utilisateurs. Quand cela se produit, un attaquant peut tirer profit des informations obtenues pour trouver des vulnérabilités. Par exemple, lorsque l'application communique avec un serveur SQL, ce dernier est susceptible de retourner une erreur. Si celle-ci remonte jusqu'à l'interface utilisateur, elle peut être utilisée par un attaquant pour déterminer si le logiciel est sensible aux injections SQL et pour extraire des données qui ne lui sont pas destinées. Dans l'exemple de la fonction de recherche, on peut entrer la saisie suivante :

```
' UNION SELECT * FROM sys.tables --
```

Si les erreurs ne sont pas correctement gérées, un message comme celui-ci peut apparaître : 4

On remarque deux choses importantes ici : l'application utilise la saisie de l'utilisateur pour générer dynamiquement du SQL qui sera exécuté par la base (injection possible), et le serveur nous indique que les nombres de champs des tables `Members` et `sys.tables` diffèrent. Ces informations suffisent à motiver un attaquant qui tentera d'autres saisies et s'appuiera sur les retours de la base SQL pour trouver le nombre de paramètres et leurs types respectifs :

```
' UNION SELECT NEWID(), name, name, 'false' FROM sys.tables --
```

Il obtiendra de cette manière les informations qu'il désire : 5

D'une manière ou d'une autre, toutes les erreurs doivent être traitées et dans tous les cas le développeur doit contrôler le message d'erreur qui sort du système.

4 Exécuter le code avec des droits trop élevés

Pour manipuler les différentes ressources dont il a besoin (système de fichiers, base de données, etc.), un programme doit s'exécuter avec certains droits. Une application telle qu'un réseau social par exemple a besoin d'un compte sur un serveur SQL pour pouvoir afficher la liste des membres ou créer de nouveaux comptes. Pour des raisons pratiques et pour simplifier les développements, il n'est pas rare d'utiliser les identifiants du compte administrateur qui a les « pleins pouvoirs » sur la base de données. Le risque que pose ce type de configuration est de permettre à l'assaillant prenant le contrôle de l'application web de manipuler des ressources en principe réservées aux administrateurs.

Dans l'exemple de la partie sur la gestion d'erreurs, nous avons vu comment un attaquant pouvait extraire des informations sur le schéma de base de données à l'aide de la requête :

```
' UNION SELECT NEWID(), name, name, 'false' FROM sys.tables --
```

Si une telle attaque fonctionne, c'est que le compte utilisé par l'application web a les droits de lecture sur le schema `sys` contenant les informations relatives à la structure de la base. Pourtant, le site

3

Résultats pour votre recherche: <script language="javascript">alert('xss');</script>

5

Résultats pour votre recherche: ' UNION SELECT NEWID(), name, name, 'false' FROM sys.tables --

1. Accounts ACCOUNTS
2. Dominique MONIER
3. Dominique DU VERGER
4. Dominique MARTIN
5. Members MEMBERS
6. Hobbies HOBBIES
7. Relations RELATIONS
8. Dominique DUPOND
9. Dominique DURANT

4

Server Error in '/' Application.

All queries combined using a UNION, INTERSECT or EXCEPT operator must have an equal number of expressions in their target lists.

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

Exception Details: System.Data.SqlClient.SqlException: All queries combined using a UNION, INTERSECT or EXCEPT operator must have an equal number of expressions in their target lists.

Source Error:

```
Line 28:                context.Members.SqlQuery("SELECT * FROM Members WHERE FirstName Like '%" + query + "';");
Line 29:
Line 30:            var members = result.ToList();
Line 31:
Line 32:            return context.Members.Where(m => m.FirstName == query)
```

Source File: C:\Users\Philippe\Documents\Projects\Top10Mistakes\Top10Mistakes\Controllers\SearchController.cs **Line:** 30

Stack Trace:

```
[SqlException (0x80131904): All queries combined using a UNION, INTERSECT or EXCEPT operator must have an equal number of expressions in their target lists.]
   System.Data.SqlClient.SqlConnection.OnError(SqlException exception, Boolean breakConnection, Action`1 wrapCloseInAction) +212
   System.Data.SqlClient.SqlInternalConnection.OnError(SqlException exception, Boolean breakConnection, Action`1 wrapCloseInAction) +81
   System.Data.SqlClient.TdsParser.ThrowExceptionAndWarning(TdsParserStateObject stateObj, Boolean callerHasConnectionLock, Boolean asyncClose) +631
```

a seulement besoin d'effectuer des recherches sur la table des membres (que l'on nommera ici Members).
Le ***principe de séparation des privilèges*** consiste à ne donner à un acteur demandant l'accès à une ressource que les droits dont il a besoin pour effectuer son travail, sans plus. En créant un compte dédié à l'application web, il est possible de limiter les droits aux opérations nécessaires. Dans notre exemple, le compte utilisé n'aura que les droits de lecture sur les tables `Members`.

```
CREATE LOGIN MyApplicationUser
  WITH PASSWORD = 'mysafepassword'
GO
CREATE USER MyApplicationUser
  FOR LOGIN MyApplicationUser
  WITH DEFAULT_SCHEMA = [dbo]
GO
GRANT SELECT ON dbo.Members TO MyApplicationUser;
DENY SELECT, VIEW DEFINITION ON SCHEMA::[sys] TO [MyApplicationUser];
DENY SELECT, VIEW DEFINITION ON SCHEMA::[INFORMATION_SCHEMA] TO [MyApplicationUser]
```

De cette manière, l'attaquant n'a plus de moyen simple de connaître le schéma de la base de données. 6
L'injection est toujours présente, le message d'erreur donne toujours trop d'information, mais l'attaquant ne peut plus reverser le design de la base et les dégâts sont limités.

Note

Dans une application réelle, les droits seraient plutôt donnés sur un ensemble de tables. Nous simplifions ici les choses pour nous concentrer sur les principes de sécurité.

5 Se contenter d'une protection côté client

Sur les architectures client-serveur, la menace est souvent vue du côté du client. En tant que développeurs, nous pensons naturellement à la donnée qui va être saisie dans l'interface utilisateur et nous y plaçons les contrôles nécessaires pour garantir le bon fonctionnement du programme. La supposition de départ est ici que tout le monde utilisera l'application avec le client que nous avons développé. L'attaquant a quant à lui un regard bien différent : s'il se rend compte de limitations imposées par ce client, il n'hésitera pas à s'adresser directement au serveur, contournant de cette manière tout protection placée par le programmeur. Reprenons le cas de notre attaque d'injection SQL. La saisie de :

```
' UNION SELECT NEWID(), name, name, 'false' FROM sys.tables --
```

pourrait être bloquée par un contrôle client du type :

```
<script language="javascript">
    var btnSubmit = document.getElementById('submit');
    btnSubmit.onclick = function (e) {
        var txtQuery = document.getElementById('query');
        var input = txtQuery.value;
        if (input && input.match(/[^A-Za-z'-]/)) {
            alert('Saisie incorrecte');
            return false;
        }
    };
</script>
```

Le code Javascript ci-dessus va empêcher l'envoi du formulaire lorsque la saisie contient autre chose que des lettres, apostrophes et traits d'union.
Malgré cela, si l'attaquant effectue la requête http suivante à l'aide d'un client type cURL ou Postman,

```
GET /Search/Search?query=' UNION SELECT NEWID(), name, name, 'false' FROM sys.tables -- HTTP/1.1
```

=> l'entrée dangereuse arrive au serveur. La seule ligne de défense du programme a été contournée : l'application reste vulnérable.
Le ***principe de défense en profondeu***r est un des piliers de la sécurité. Il consiste à placer plusieurs mécanismes de protection les uns derrière les autres de sorte que, lorsqu'un attaquant parvient à franchir une ligne de défense, il doit faire face à la suivante. L'idée est qu'une protection n'a pas pour seul but de stopper l'ennemi : elle sert aussi à l'affaiblir, à lui faire perdre son intérêt pour ce qu'il est venu chercher. Si l'effort nécessaire au succès de l'attaque dépasse les bénéfices potentiels, l'assaillant déclarera probablement forfait. Dans notre cas, il faudra cumuler les protections. En plus d'effectuer une validation client, on fera une validation côté serveur (défaut 1). La donnée sera ensuite assainie avant d'être envoyée au serveur SQL (défaut 2), et la requête sera exécutée avec des droits restreints (défaut 4). On éliminera ainsi le risque d'injection SQL.

6 Utiliser un seul contrôle avant de permettre l'accès aux ressources

Certaines ressources et opérations doivent être protégées. Trop souvent, le contrôle d'accès ne se fait pas au niveau de la ressource ni de l'opération, mais au niveau de l'application.
L'utilisateur doit entrer ses identifiants pour accéder aux différents services offerts par le programme. Une fois cette étape franchie, des contrôles doivent être effectués de sorte qu'un utilisateur ne puisse pas accéder aux ressources qui ne sont pas les siennes. Ces vérifications sont pourtant fréquemment oubliées, ce qui est à l'origine d'une vulnérabilité très répandue : la référence directe non sécurisée à un objet.
Imaginons un site rendant accessibles les ressources d'un utilisateur via URL, comme par exemple :
https://monsite.com/mesdocuments/319a3f89-81e2-411c-b19d-f15b6722a998

Ici, 319a3f89-81e2-411c-b19d-f15b6722a998 représente l'identifiant en base de données du document à consulter. Sans contrôle approprié, toute personne pouvant utiliser le système et connaissant l'URL peut accéder à l'information.
Sauf pour les documents publics, quelqu'un qui connaît simple-

6

Server Error in '/' Application.

The SELECT permission was denied on the object 'tables', database 'mssqlsystemresource', schema 'sys'.

Description: An unhandled exception occurred during the execution of the current web request. Please review the stack trace for more information about the error and where it originated in the code.

Exception Details: System.Data.SqlClient.SqlException: The SELECT permission was denied on the object 'tables', database 'mssqlsystemresource', schema 'sys'.

ment une URL ne doit pas être en mesure de consulter la ressource qui se cache derrière, même s'il s'agit d'un utilisateur légitime. Un contrôle doit être fait au niveau de l'action, garantir que la personne a le droit d'accéder au document en plus de pouvoir utiliser le système.

```
public Document Get(Guid documentId)
{
  if(User.CanAccess(documentId))
  {
    return _context.Documents.Single(d => d.Id == documentId);
  }
  throw new InvalidOperationException("Accès refusé");
}
```

7 Stocker les mots de passe avec une protection insuffisante

Si un assaillant parvient à mettre la main sur le contenu de la base de données, il est impératif de minimiser les dégâts qu'il peut faire avec. L'un des pires scénarios est de voir les mots de passe des utilisateurs disponibles sur internet. La méthode employée pour stocker ces mots de passe est donc primordiale. Pourtant, c'est un point négligé dans de nombreux logiciels.

La plus grande erreur consiste à enregistrer les mots de passe en clair dans une table. Dans ces conditions, dès que les données sont extraites du serveur, l'attaquant a terminé son travail.

Une autre erreur est de stocker le mot de passe chiffré. Même si cette solution est meilleure que la précédente, le problème est reporté sur la protection de la clé utilisée pour le chiffrement. Si l'attaquant la trouve, il peut déchiffrer les mots de passe.

Idéalement, l'application n'a pas connaissance du mot de passe. Elle utilise donc une fonction de hachage et sauvegarde les empreintes à la place des mots de passe en clair. Là encore, les possibilités de commettre une faute sont nombreuses. Les fonctions très répandues, comme MD5 ou SHA-1 sont vulnérables aux attaques par brute force, par dictionnaire ou par Rainbow table. Les outils tels que hashcat permettent de cracker des mots de passe en un temps record.

Pour compliquer la tâche, il faut employer un sel, valeur arbitraire qui sera ajoutée au mot de passe fourni à la fonction de hachage. Cependant, les ordinateurs et les outils devenant de plus en plus performants, les sels ne sont plus suffisants.

On privilégiera donc les fonctions de dérivation de clé, qui ajoutent à la fonction de hachage et au sel un nombre d'itérations. Lors de la première itération, mot de passe et sel sont fournis à la fonction. Le produit de cette opération est à son tour passé à la fonction de dérivation, et on répète ce procédé jusqu'à atteindre le nombre d'itérations voulu, celui-ci pouvant être augmenté périodiquement pour faire face au gain de puissance des armes de l'adversaire.

8 Transmettre les données sensibles à travers un canal non sécurisé

Comme nous avons déjà pu le voir, la sécurité de notre application n'est pas limitée au code que nous produisons. L'une des cibles favorites des attaquants est le canal de communication entre l'application et les systèmes externes. Par ce biais, il espère intercepter du contenu pouvant lui servir, comme des mots de passe, des numéros de cartes bleues, etc.

De nombreux sites internet offrent des sections réservées aux abonnés. Pour y accéder, il est nécessaire d'entrer un login et un mot de passe dans un formulaire d'authentification. Il est fréquent de voir ces formulaires servis via HTTP, un protocole non sécurisé. Dans ce cas, rien n'empêche un pirate de se placer entre le navigateur et le serveur. Il pourra de cette manière récupérer le mot de passe de la victime au moment où elle soumettra le formulaire.

Pour les échanges critiques, il convient d'utiliser un canal sécurisé. En forçant l'usage du protocole HTTPS dans le cas d'un site web, l'attaquant n'a plus la possibilité de voir l'information dans le trafic capturé. Il faudra de la même façon veiller à adopter un canal chiffré aussi bien pour transmettre les pages qui récoltent les données que pour les URLs cibles. Autrement, rien ne garantit que le formulaire de saisie de l'information sensible n'ait pas été modifié (pour envoyer les données directement au hacker par exemple).

D'une manière générale, les certificats auto-signés sont également à proscrire. Ils peuvent être générés facilement par un attaquant qui pourra se placer entre le navigateur et le serveur tout comme s'il n'y en avait pas. Notons toutefois que les navigateurs modernes font un bon travail pour avertir les internautes lorsqu'un site utilise un certificat douteux.

9 Utiliser un système d'authentification défaillant

Nous avons vu plusieurs défauts de programmation qui affectent la sécurité de nos applications. Ceux-ci s'accumulent fréquemment autour d'une fonctionnalité pourtant critique : l'authentification.

Les possibilités pour affaiblir un système d'authentification sont nombreuses. Le canal de communication ou le stockage des mots de passe peuvent être mal protégés. On trouve aussi des politiques dommageables sur les formats de mots de passe, comme celles de certains sites bancaires exigeant des codes à 6 chiffres. Parfois, c'est la volonté de faciliter la vie de l'utilisateur qui est en cause.

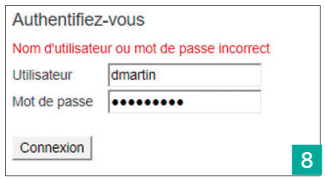
Les systèmes d'authentification classiques comportent deux champs, l'un permettant la saisie d'un nom d'utilisateur ou d'un email, l'autre permettant la saisie d'un mot de passe.

Lorsque la combinaison (nom d'utilisateur, mot de passe) est valide, l'utilisateur est « authentifié » par le système. Il peut alors accéder à différentes fonctionnalités, selon les autorisations qui lui sont accordées.

Quand la combinaison est incorrecte en revanche, l'accès au système est refusé et un message d'erreur est affiché. Les deux de ces messages suivants sont très répandus 7

Le développeur pense ici bien faire et aide l'utilisateur à s'authentifier correctement. En revanche, il aide aussi un attaquant à faire une attaque par énumération, dont le principe est simple : On entre un nom d'utilisateur ou une adresse email et un mot de passe au hasard, on valide et on regarde le message d'erreur. Le message Mot de passe incorrect nous indique qu'il existe un utilisateur valide avec ce nom ou cette adresse email.

Un fois que l'attaquant se sera constitué un répertoire d'utilisateurs, il pourra tenter de deviner les mots de passe associés en uti-



lisant des mots de passes courants ou des mots du dictionnaire. Plus son répertoire sera grand, plus il aura de chances de succès. Pour éviter ce genre d'attaques (énumération + dictionnaire), on peut renvoyer un message identique dans les 2 cas. « *Nom d'utilisateur ou mot de passe incorrect* ». Ainsi, l'attaquant doit à chaque essai trouver un nom d'utilisateur valide et un mot de passe valide, ce qui change totalement la nature du problème (et le rend la tâche bien plus difficile). 8

10 Supposer que l'application s'exécute dans un contexte mono-thread

Lorsqu'on développe une application à destination d'un grand nombre d'utilisateurs, on teste la plupart du temps notre code seul. Dans ces conditions, il est facile d'oublier que certaines ressources utilisées par notre programme vont être partagées.

Les erreurs qui peuvent se produire sont nombreuses quand plusieurs demandeurs, qu'il s'agisse d'utilisateurs ou de threads, tentent d'accéder à la même ressource en même temps. Leurs conséquences sont tout aussi variées, allant du simple bug à peine perceptible à l'indisponibilité totale de l'application. Ces bugs peuvent être exploités par un utilisateur malintentionné.

Prenons l'exemple d'une application de gestion de cagnotte. Elle permet de créer des cagnottes, les créditer, les débiter, les fermer, etc. Les cagnottes sont partagées entre plusieurs utilisateurs qui peuvent effectuer les opérations citées en même temps. Le code suivant, *qui semble correct à première vue*, pose en fait un problème de sécurité.

```
public void Debiter(Guid participantId, Guid cagnotteId, decimal montant)
{
  var cagnotte = _context.Cagnottes.Single(c => c.Id == cagnotteId);
  if (cagnotte.Total >= montant)  // contrôle (1)
  {
    var participant = _context.Participants.Single(p => p.Id == participantId);
    participant.Solde += montant;
    cagnotte.Total -= montant; // mise à jour (2)
    _context.SaveChanges();   // retrait effectif (3)
  }
}
```

Exécutée individuellement, la méthode semble faire son travail : débiter la cagnotte au profit d'un participant. Quand on prend en compte le fait que plusieurs utilisateurs peuvent effectuer des opérations en même temps, on se rend compte du problème : il est possible de retirer plus que le montant total de la cagnotte. Il suffit pour ce faire qu'un premier utilisateur demande le retrait de la totalité. A l'exécution du contrôle (1) le montant demandé est disponible. L'application va donc permettre le débit. Si entre le moment où l'accord est donné (1) et le retrait effectif (3), un second contrôle est effectué (1) pour une autre demande de débit du montant total, le logiciel pensera que la cagnotte contient les fonds nécessaires et permettra le second débit. Au moment où le retrait effectif de la première opération à lieu, la cagnotte est vide. Lorsque vient le deuxième retrait effectif, la cagnotte passe dans le rouge. Un utilisateur malveillant qui remarquerait ce bug pourrait l'utiliser à son avantage en émettant lui-même deux demandes de débit au même moment.

Deux opérations qui affectent une ressource partagée ne doivent pas pouvoir se dérouler en même temps. Le programmeur doit faire preuve de vigilance et employer les techniques adaptées à ces situations (mutex ou autre).

En conclusion

Bien qu'il existe beaucoup d'autres défauts de sécurité, nous venons de parcourir 10 erreurs très répandues qui sont à l'origine d'un grand nombre de failles. En les supprimant de votre code et en veillant à respecter les principes fondamentaux de la sécurité logicielle, vous vous assurez de donner « du fil à retordre » à ceux qui auraient la volonté d'attaquer vos applications, et de ne plus leur faciliter la vie. •

Christophe PICHAUD
Consultant sur les technologies Microsoft
christophepichaud@hotmail.com |
www.windowscpp.net

NeOS-SDI makes IT work

Utiliser shared_ptr<T> en C++ pour la gestion des ressources

Le C++ n'a aucune leçon à recevoir des langages managés (C#, Java) pour la gestion automatique du cycle de vie des ressources car nous avons depuis TR1, les « smart pointers » ou pointeurs intelligents. Nous avons à notre disposition des templates comme std::shared_ptr<T> et std::unique_ptr<T>, et il est possible de nous passer de delete dans notre code, ceci sans sacrifier le contrôle et la performance. Le C++ est supérieur à tous ces langages soi-disant « safe and secure » qui possèdent une machinerie en arrière-plan qui compense les opérations que leurs développeurs n'ont pas à faire.

Ce mois-ci, je vous propose l'adaptation d'un paragraphe du livre de Scott Meyers « Effective Modern C++ », « 42 specific ways to improve your use of C++11 and C++14 ». Je vous propose donc de commenter l'élément 19 du livre qui se nomme Item 19 : utilisation du shared_ptr pour la gestion des ressources partagées.

Présentation de la mécanique

std::shared_ptr est le moyen en C++ 11 de combiner le meilleur de ces mondes. Un objet accédé via std::shared_ptr possède un cycle de vie géré par ses pointeurs au travers de de la propriété partagée. Aucun std::shared_ptr ne possède l'objet. A la place, tous les std::shared_ptrs qui pointent dessus s'entendent pour que sa destruction soit faite quand il n'est plus nécessaire.

Quand le dernier std::shared_ptr pointant sur un objet arrête de pointer dessus (parce que le std::shared_ptr est détruit ou pointe sur un objet différent), ce std::shared_ptr détruit l'objet sur lequel il pointe. Comme avec le garbage collector, les clients ne sont pas obligés de gérer eux-mêmes le cycle de vie des objets pointés, mais avec le mécanisme des destructeurs, la destruction de ces objets est déterministe.

Un std::shared_ptr peut dire que c'est le dernier à pointer sur une ressource en consultant le compteur de références de la ressource, une valeur associée avec la ressource qui garde trace de combien de std::shared_ptr pointent dessus. Les constructeurs de std::shared_ptr incrémentent ce compteur, les destructeurs std::shared_ptr décrémentent le compteur, et les opérateurs de copie font les deux.

(si sp1 et sp2 sont des std::shared_ptr sur différents objets, l'affectation "sp1 = sp2;" modifie sp1 de telle manière qu'il pointe sur l'objet pointé par sp2. L'effet de cette affectation et que le compteur de référence pour cet objet originalement pointé par sp1 est décrémenté, pendant que pour l'objet pointé par sp2 est incrémenté.) Si un std::shared_ptr voit son compteur de référence à zéro après une opération de décrémentation, plus aucun std::shared_ptrs ne pointe sur la ressource donc le std::shared_ptr le détruit.

Les détails

L'existence du compteur de référence a des implications de performance :

- la taille d'un **std::shared_ptr est double comparée à celle d'un pointeur raw,** parce qu'il contient un pointeur explicite vers la ressource et aussi vers le compteur de référence.
- **La mémoire pour le compteur de référence doit être allouée dynamiquement**. Le compteur de référence est associé avec l'objet pointé mais les objets pointés n'en savent rien. Ils n'ont aucune place pour stocker un compteur de référence. (N'importe quel objet – même les types built-in – peuvent être gérés par std::shared_ptr.) La figure explique que le coût de l'allocation dynamique est évitée quand le std::shared_ptr est créé par std::make_shared, mais il est des cas où std::make_shared ne peut pas être utilisé. Dans ce cas, le compteur de référence est stocké en données allouées dynamiquement.
- **Les incrémentations et décrémentions du compteur de référence doivent être atomic,** parce qu'il peut y avoir simultanément des lecteurs et écrivains dans différents threads.

Par exemple, un std::shared_ptr qui pointe sur une ressource dans un thread peut exécuter son destructeur (décrémentation du compteur de référence pour la ressource qu'il pointe), pendant que dans un thread différent, un std::shared_ptr sur le même objet peut être copié (et donc incrémenter le même compteur de référence). Les opérations atomic sont typiquement plus lentes que les opérations non-atomic, mais vous devez savoir que même sur une telle opération, la lecture et l'écriture d'un compteur de référence a un prix coûteux.

Votre curiosité s'est-elle éveillée quand je vous écris qu'un constructeur de std::shared_ptr ne fait qu'incrémenter le compteur de référence pour l'objet sur lequel il pointe ?

Créer un std::shared_ptr qui pointe sur un objet fait plus que pointer sur un objet, donc pourquoi est-ce qu'on ne peut pas incrémenter le compteur de référence à tous les coups ?

La construction par déplacement (move construction), c'est la raison. Faire une construction de déplacement d'un std::shared_ptr vers un autre std::shared_ptr positionne la source std::shared_ptr à null et ça veut dire que le vieux std::shared_ptr arrête de pointer sur la ressource au moment où le nouveau std::shared_ptr démarre. En résultat, aucune manipulation de compteur de référence n'est requise. Le déplacement de std::shared_ptr est plus rapide que la copie car la copie requièrt l'incrémentation du compteur de référence, mais le déplacement non.

C'est vrai pour l'affectation en construction, donc la construction par déplacement est plus rapide que la copie par construction et l'affectation de déplacement est plus rapide que l'affectation de copie. Comme std::unique_ptr, std::shared_ptr utilise delete comme son mécanisme de destruction de ressource par défaut, mais il supporte aussi les deleters spécifiques. Le design de cette fonctionnalité n'est cependant pas tout à fait standard pour un std::unique_ptr.

Pour std::unique_ptr, le type du deleter fait partie du type de smart pointer. Pour std::shared_ptr, ce n'est pas :

```
auto loggingDel = [](Widget *pw) // custom deleter
{
makeLogEntry(pw);
delete pw;
};
std::unique_ptr<        // deleter type is
Widget, decltype(loggingDel)    // part of ptr type
> upw(new Widget, loggingDel);
std::shared_ptr<Widget>    // deleter type is not
spw(new Widget, loggingDel); // part of ptr type
```

Le design de std::shared_ptr est plus flexible. Considérons deux std::shared_ptr<Widget>s, chacun avec un custom deleter d'un type différent (parce que les custom deleters sont spécifiés via des lambdas):

```
auto customDeleter1 = [](Widget *pw) { … };  // custom deleters,
auto customDeleter2 = [](Widget *pw) { … }; // each with a
// different type
std::shared_ptr<Widget> pw1(new Widget, customDeleter1);
std::shared_ptr<Widget> pw2(new Widget, customDeleter2);
```

Parce que pw1 et pw2 ont le même type, ils peuvent être placés dans un container d'objets de ce type :

```
std::vector<std::shared_ptr<Widget>> vpw{ pw1, pw2 };
```

Ils peuvent aussi être assignés ou passés à une fonction qui prend un paramètre de type std::shared_ptr<Widget>. Aucune de ces possibilités ne peuvent être effectuées avec des std::unique_ptrs qui diffèrent dans le type de leur custom deleters parce que le type du custom deleter modifie le type du std::unique_ptr.

Une autre différence pour std::unique_ptr, spécifiant un custom deleter ne change pas la taille d'un objet std::shared_ptr. Quel que soit le deleter, un objet std::shared_ptr vaut 2 pointeurs en taille. C'est super mais bon… Les custom deleters peuvent être des objets fonctions et peuvent contenir n'importe quelle quantité de data. Ça veut dire qu'ils peuvent être larges. Comment un std::shared_ptr qui contient un custom deleter de taille arbitraire fait-il sans utiliser plus de mémoire ? Il ne peut pas. Il doit utiliser plus de mémoire.

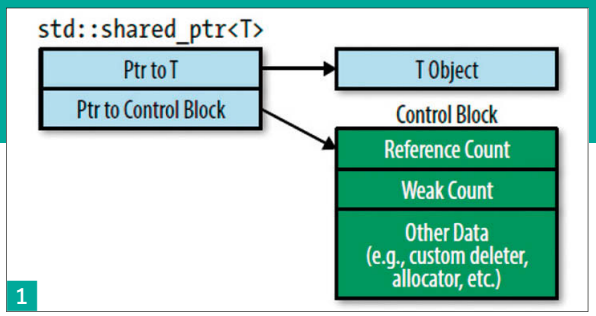
Cependant, cette mémoire ne fait pas partie de l'objet std::shared_ptr. C'est sur le cas où, si le créateur du std::shared_ptr prend avantage du support pour les custom allocators, c'est là que la mémoire gérée par l'allocateur est localisée. On remarque qu'un objet std::shared_ptr contient un pointeur vers le compteur de référence pour l'objet sur lequel il pointe. C'est vrai mais plus largement, le compteur de référence fait partie d'une structure de données plus large qui est le « control block ». Il y a un control block pour chaque objet géré par un std::shared_ptr.

Si spécifié, le control block contient en plus du compteur de référence, une copie du custom deleter si spécifié, un custom allocator si spécifié et aussi d'autres informations comme un compteur de référence. Voici à peu près à quoi cela ressemble : 1

Un objet control block est créé par la fonction qui crée le premier std::shared_ptr vers un objet. Mais ce n'est pas évident, donc les règles suivantes pour la création du control block sont :

- **std::make_shared en crée toujours un.** C'est le meilleur endroit car on démarre ;
- **un control block est créé quand std::shared_ptr est construit à partir d'un unique-ownership pointer (std::unique_ptr ou std::auto_ptr).**

Les pointeurs unique-ownership n'utilisent pas de control blocks. De plus,



1

lors de sa construction, le std::shared_ptr prend la possession de l'objet et le uniqueownership pointeur est mis à null.

- **Quand un constructeur de std::shared_ptr est appelé avec un raw pointeur, il crée le control block.**

Si vous voulez créer un std::shared_ptr à partir d'un objet qui a déjà un control block, vous allez probablement passer un std::shared_ptr ou un std::weak_ptr comme argument au constructeur et non un raw pointeur.

Les constructeurs de std::shared_ptr qui prennent un std::shared_ptr ou un std::weak_ptr en argument de constructeur ne créent pas de nouveau control block car ils s'appuient sur les smart pointers passés en argument pour pointer sur un control block.

La conséquence de ces règles est que construire plus d'un std::shared_ptr depuis un simple raw pointer vous amène droit dans une partie au comportement indéfini car il y aura plusieurs control blocks. Cela impliquerait la destruction multiple de l'objet et c'est mauvais ! Exemple de mauvais code :

```
auto pw = new Widget; // pw is raw ptr
…
std::shared_ptr<Widget> spw1(pw, loggingDel); // create control
// block for *pw
…
std::shared_ptr<Widget> spw2(pw, loggingDel); // create 2nd
// control block
// for *pw!
```

La création d'un pointeur raw vers un objet alloué dynamiquement est mauvaise car cela va à l'encontre des conseils précédemment évoquées : préférer les smart pointers aux raw pointeurs.

La ligne qui crée pw est foireuse mais au moins, elle ne crée pas de comportement indéfini. Maintenant que le constructeur de spw1 est appelé avec un raw pointeur, il crée un control block (et aussi un compteur de référence). Dans ce cas, c'est *pw. Le problème c'est que spw2 est appelé avec le même raw pointeur et donc il y a création d'un control block pour *pw. *pw possède maintenant 2 compteurs de référence susceptibles de passer à zéro et d'essayer de détruire *pw deux fois ! La seconde destruction est responsable d'un comportement indéfini qui sera certainement un plantage en règle. Il y a deux leçons au regard de std::shared_ptr utilisées ici. Premièrement, essayer de passer un raw pointer au constructeur d'un std::shared_ptr. L'alternative habituelle est d'utiliser std::make_shared mais dans notre exemple, nous utilisons un custom deleter et ce n'est pas possible avec std::shared_pr. Deuxièmement, si vous devez passer un raw pointer au constructeur de std::shared_ptr, passez le résultat du new directement au lieu de créer une variable raw pointer.

```
std::shared_ptr<Widget> spw1(new Widget, // direct use of new
loggingDel);
```

Créons-nous un second std::shared_ptr à partir du même raw pointer ? Non, à la place nous allons créer spw2 en fournissant spw1 comme initialisation dans le constructeur par copie et cela ne pose aucun problème :

```
std::shared_ptr<Widget> spw2(spw1);    // spw2 uses same
// control block as spw1
```

Dans ce cas, on va avoir plusieurs control blocks qui protègent le même raw pointer. Prenons un exemple pour que le programme gère des Widgets à traiter :

```
std::vector<std::shared_ptr<Widget>> processedWidgets;
```

Voici la classe Widget:

```
class Widget {
public:
…
void process();
…
};
```

Voici la méthode Process :

```
void Widget::process()
{
…                              // process the Widget
processedWidgets.emplace_back(this);          // add it to list of
}                          // processed Widgets;
// this is wrong!
```

C'est une erreur de passer le this, pas d'utiliser emplace_back. Pour gérer le this, il faut dériver de std:: enable_shared_from_this.

```
class Widget: public std::enable_shared_from_this<Widget> {
public:
…
void process();
…
};
```

std::enable_shared_from_this définit une fonction membre qui crée un std::shared_ptr pour l'objet courant sans dupliquer les control blocks. Voici une implémentation de process :

```
void Widget::process()
{
// as before, process the Widget
…
// add std::shared_ptr to current object to processedWidgets
processedWidgets.emplace_back(shared_from_this());
}
```

En interne, shared_from_this regarde le control block pour l'objet courant et il crée un nouveau std::shared_ptr qui pointe sur ce control block. Le design est fonction de l'objet courant qui possède un control block. Pour ce cas-là, il faut qu'il existe un std::shared_ptr qui pointe sur l'objet courant.

Si aucun std::shared_ptr n'existe, le comportement est indéfini. En gros, c'est le plantage assuré.

Avant que un std::shared_ptr pointe sur l'objet, les classes qui héritent de std::enable_shared_from_this déclarent leurs constructeurs private et ont des clients qui créent des objets en appelant une fonction factory qui retourne un std::shared_ptrs. Widget, par exemple, pourrait ressembler à ça :

```
class Widget: public std::enable_shared_from_this<Widget> {
public:
// factory function that perfect-forwards args
// to a private ctor
template<typename... Ts>
static std::shared_ptr<Widget> create(Ts&&... params);
…
void process(); // as before
…
private:
… // ctors
};
```

Maintenant, nous savons quand il faut limiter le nombre de block controls pour un std::shared_ptr.

Un control block vaut en taille quelques octets, bien que les custom deleters et allocators peuvent les rendre plus larges. L'implémentation est en réalité plus complexe que l'on ne pense. Il y a de l'héritage et même une fonction virtuelle qui est utilisée pour le pointeur sur objet soit proprement libéré. On est maintenant conscient que std::shared_ptr apporte le coût d'une machinerie complexe.

Après avoir pris connaissance des allocations dynamiques des controls blocks, des deleters et allocators, de la machinerie de la fonction virtuelle, des manipulations de compteurs de référence, vous vous dites qu'il y a du monde… Ne prenez pas peur.

Ce n'est pas la meilleure solution pour chaque problème de gestion de ressources mais pour la fonctionnalité qu'ils fournissent, les std::shared_ptrs offrent un coût raisonnable.

Dans des conditions ou le deleter par défaut et l'allocateur par défaut sont utilisés et que le std::shared_ptr est créé par std::make_shared, le control block n'est que de 3 words en taille.

Déréférencer un std::shared_ptr n'est pas plus coûteux que de déréférencer un raw pointer.

Réaliser une opération requiert une manipulation du compteur de référence (comme l'utilisation de constructeur de copie ou l'affectation par copie ou la destruction) via une ou deux opérations atomic. Ce sont des instructions simples peu coûteuses. La machinerie de fonction virtuelles dans le control block est utilisée une fois par objet géré par std::shared_ptrs quand l'objet est détruit.

La plupart du temps, utiliser std::shared_ptr est préférable pour gérer le cycle de vie d'un objet partagé à la main. Si le partage n'est pas nécessaire, vous pouvez utiliser std::unique_ptr. Il est possible de passer de l'un à l'autre mais uniquement dans le sens std::unique_ptr vers std::shared_ptr.

Note

En échange de ces coûts modestes, vous avez un management de cycle de vie automatique des ressources allouées dynamiquement.

A retenir

- Les std::shared_ptr offrent un système qui se rapproche du garbage collector pour le management du cycle de vie des ressources partagées.
- Comparé à std::unique_ptr, les objets std::shared_ptr sont deux fois plus gros car il y a le control block et les manipulations atomic du compteur de références.
- La destruction de ressource par défaut est via delete mais les custom deleters existent. Le type de deleter n'a aucun effet sur le type std::shared_ptr.
- Eviter de créer des std::shared_ptr à partir de variables de type raw pointers.

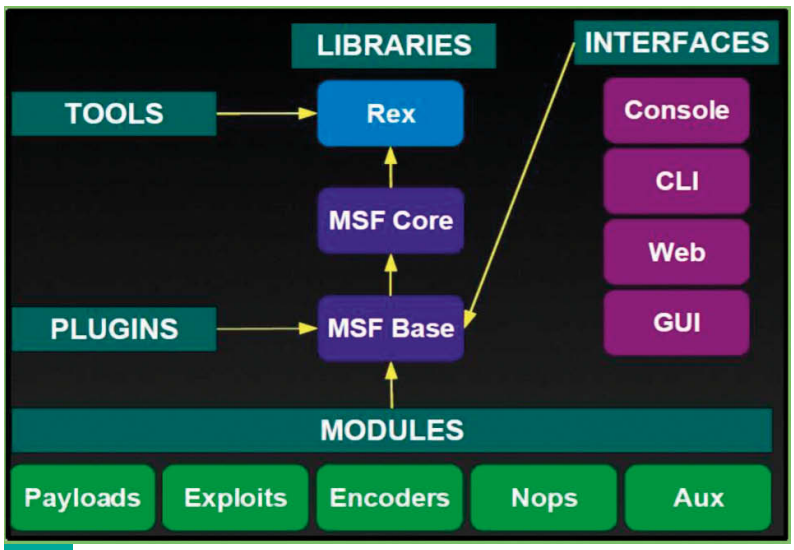
Conclusion

En tant qu'utilisateur du std::shared_ptr<T>, tout est simple. En interne, c'est plus compliqué et remarquablement bien fait. Pouvoir se passer du new dans un code est une fonctionnalité géniale. Tout est automatique et pourtant on est en C++. Une fois de plus : « Power and Performance ». •

Erik Lenoir
Responsable du pôle Sécurité chez Zenika

Metasploit : en 2017 ça donne quoi ?

Metasploit est sûrement le « must have » pour tout auditeur en sécurité, et il ne cesse de se renouveler. Beaucoup de documentations existent sur le Web concernant cet outil mais peu sont à jour et rentrent vraiment dans le cœur du framework.



1 *Architecture de Metasploit*

« Metasploit Pen Testing Tool », est un projet open-source (sous Licence BSD modifiée) pour la sécurité des systèmes informatiques. Il a pour objectif de fournir des informations sur les vulnérabilités de systèmes informatiques, d'aider à la pénétration et au développement de signatures pour les IDS. Le plus connu des sous-projets est le Metasploit Framework, un outil pour le développement et l'exécution d'exploits (logiciels permettant d'exploiter à son profit une vulnérabilité) contre une machine distante. Créé à l'origine en langage de programmation Perl, Metasploit Framework a été complètement réécrit en langage Ruby. Le plus notable est la publication de certains des exploits les plus techniquement sophistiqués auprès du public. C'est un outil très puissant pour les chercheurs en sécurité travaillant sur les potentielles vulnérabilités de systèmes informatiques.

Metasploit peut être utilisé par les administrateurs pour tester la vulnérabilité des systèmes informatiques afin de les protéger, ou par les pirates et les script kiddies (« gamins à script ») à des fins de piratage. Comme la plupart des outils de sécurité informatique, Metasploit peut être utilisé à la fois de manière légale et à la fois pour des activités illégales. Le fait que Metasploit ait émergé en tant que plateforme de développement dans la sécurité, a conduit, ces derniers temps, à la publication de vulnérabilités logicielles souvent accompagnées d'un module d'exploitation pour Metasploit pour ces dernières, afin de mettre en évidence l'exploitabilité, le risque et les mesures de prévention contre ces bogues particuliers. Metasploit 3.0 (en langage Ruby) a également commencé à inclure des outils de fuzzing, pour découvrir des vulnérabilités de logiciels en premier lieu, plutôt que de simplement être fait pour l'exploitation de celles-ci. De plus, une des forces de Metasploit est sa capacité à interagir avec d'autres outils comme nmap, sqlmap, John The Ripper, le tout centralisé dans la console du framework.

Et sous le capot alors ? [1]

Le Framework embarque des bibliothèques qui aident à l'écriture rapide d'exploit, afin de ne pas avoir à se soucier d'écrire du code pour des tâches basiques telles qu'une requête HTTP ou l'encodage d'un payload.

Il y a trois librairies importantes dans le Framework :

- Rex – la librairie principale proposant la gestion des sockets, protocoles, transformations du texte, SSL, XOR, Base64, etc ;
- Msf::Core – fournit une API basique ;
- Msf::Base – fournit une API dite « amie ».

Ainsi, pour écrire un module basique, **seule la connaissance de l'API de Base est nécessaire**.

Comme nous pouvons le voir sur le schéma, MSF est composé de modules, c'est ce qui en fait sa grande force. Les modules peuvent être rechargés sans redémarrage de toute l'application, ce qui est très pratique pour le développement. Les principaux sont les exploits définis en tant que modules nécessitant un payload. Si ce n'est pas le cas, alors c'est un module Auxiliaire. Les payloads sont les codes exécutables à distance, c'est la charge active qui s'exécutera si tout se passe bien. Il s'agit d'un shell, d'un reverse TCP ou HTTP, etc. Les "encoders" permettent d'effectuer des actions de dissimulation sur un payload (notamment car les antivirus connaissent parfaitement les signatures des payloads Metasploit). Les nops permettent de garder une payload à une taille fixe : en effet, si nous prenons l'exemple de l'exploitation d'un Buffer Overflow, le payload doit souvent avoir une taille bien spécifique en octets (ni plus ni moins). Ainsi, si le payload initial se retrouve plus petit, on va le « bourrer » (notion de « padding ») avec des "nops" pour atteindre la taille souhaitée.

Les post sont les modules de post exploitation, c'est-à-dire qu'ils servent à effectuer une action sur une cible compromise (espionnage, collecte d'informations, etc.). En cas d'incompréhension sur ces notions, il suffit d'imaginer un missile. Notre missile est composé d'un système de propulsion et de guidage vers sa destination : il s'agit de l'exploit. Il peut embarquer une charge utile : une charge explosive, une arme chimique, etc. : il s'agit du payload. Si notre missile n'est pas « capable » d'embarquer une charge utile, c'est un module auxiliaire. Notre missile peut être détecté par sa signature thermique, on peut tenter de la masquer avec un encodeur. Ainsi, Metasploit devient un lanceur de missiles sur mesures, avec un catalogue de choix important, permettant de frapper tout adversaire, si celui-ci est sensible à un missile en particulier. Au niveau du système de fichiers, nous retrouvons les éléments suivants :

Fichier/Dossier	Signification
CONTRIBUTING.md	Guide sur « Comment contribuer »
data	Fichiers éditables et utilisés par Metasploit (wordlists…)
db	Fichiers pour la base de données
docker	Utilisation avec Docker
docker-compose.yml	Fichier Docker compose
documentation	Documentation
external	Code source et bibliothèques tierces
lib	Bibliothèques internes
modules	Modules du framework
msfconsole	Console
msfd	Daemon
msfupdate	Mise à jour du framework
msfvenom	Générateur de payload (anciennement msfpayload)
plugins	Plugins
tools	Outils en ligne de commande
Vagrantfile	Fichier Vagrant (pour une utilisation en machine virtuelle)

Les payloads

Metasploit est devenu un framework incontournable ; sur les sites comme ExploitsDB qui recensent la liste des vulnérabilités, il est très fréquent de trouver déjà le module Metasploit permettant d'exploiter cette vulnérabilité. Cela est notamment dû à la facilité d'intégration d'un module (c'est un simple fichier Ruby) et au fait que les API pour développer son propre module sont très simples d'utilisation ; dans la plupart des cas, il suffit de repartir d'un module existant et de modifier quelques lignes selon la vulnérabilité trouvée. Ainsi, la liste des payloads est immense : il y en a pour tous les goûts, les OS (MacOS, BSD, Windows, Linux, Android…), et les langages (Java, PHP, Python…).

Post exploitation

Ces modules utilisent une session/un shell et permettent d'effectuer des actions diverses et variées : extraction de données, enregistrement de frappes, capture d'écran, etc. Ces modules sont classés en fonction de leur but. Par exemple, si le module sert à la collecte de données, il va être classé dans la catégorie « gather ». S'il ajoute/modifie/supprime un utilisateur, il sera dans la catégorie « manage ». Voici la référence des catégories :

Catégorie	Description
gather	Collecte/énumération/récupération de données
gather/credentials	Vol d'informations d'identification (utilisateurs/mots de passe, etc…)
gather/forensics	Collecte d'informations forensics
manage	Modification/transformation/manipulation du système
recon	Reconnaissance et aide à l'identification d'un système, mais pas de vol de données (ce n'est pas la même chose que « gather »)
wlan	Tâches relatives aux réseaux sans fils
escalate	Cette catégorie est obsolète. Elle était utilisée pour les modules d'élévation de privilèges, mais ils ne sont plus considérés comme des modules de « post exploitation » mais plutôt comme des modules d'exploitation
capture	Écoute/surveillance pour la récupération de données (par exemple les enregistreurs de frappes)

Auxiliary

Les modules auxiliaires de Metasploit ne sont pas si différents des exploits, la différence réside uniquement dans l'absence de session à la fin d'une exécution réussie. Il existe de nombreuses catégories de modules auxiliaires, de la même façon que pour les « post » :

Catégorie	Description
admin	Modification/altération/manipulation de la machine cible
analyze	Initialement prévu pour les modules de cassage de mots de passe qui demandent un temps d'exécution conséquent
client	Initialement prévu pour les modules d'ingénierie sociale
dos	Déni de service
fuzzers	Outils de tests de données aléatoires. Les sous-répertoires déterminent le protocole
gather	Récupération/collecte/énumération de données sur une cible particulière
scanner	Tous les modules utilisant Msf::Auxiliary::Scanner
server	Serveurs pour différents protocoles/services

ROADMAP

L'équipe de Metasploit a prévu des améliorations significatives, surtout au niveau du cœur du produit.

REST

Un des premiers objectifs est de pouvoir exposer sous le format d'une interface REST toutes les commandes nécessaires au fonctionnement du framework. Cela permettra notamment de pouvoir interagir plus facilement avec les autres outils.

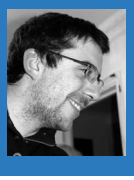
De notre point de vue, cela aura un autre avantage, celui de pouvoir plugger sa propre interface Web pour faciliter le reporting. En effet, pour avoir un reporting complet aujourd'hui, il faut obligatoirement souscrire à Metasploit Express ou Pro : l'arrivée de cette interface REST pourrait faire émerger des projets permettant la gestion graphique du framework via des applications Web.

Metasploit Aggregator

Actuellement, la gestion des sessions est faite directement en lien avec le framework. Cela pose quelques problèmes notamment pour le partage de sessions, leur monitoring et leur performance globale. Ainsi, Metasploit Aggregator est un projet initié cette année pour jouer le rôle de proxy entre la session et les listeners de Metasploit. L'objectif est d'ailleurs de mettre en place des échanges sous forme d'évènements. Cet agrégateur n'aura ainsi qu'une seule connexion vers le framework, ce qui permettra de faire basculer facilement les sessions d'un utilisateur à un autre et d'améliorer les performances globales du framework car celui-ci n'aura pas besoin de créer des threads pour gérer ces sessions.

Mais encore…

En plus de ces changements structurels importants, l'équipe aimerait avancer sur le support iOS et Android ainsi que moderniser ses payloads (notamment sur la partie Windows) et leur génération (pouvoir facilement générer depuis le framework des payloads en C, .NET, Java…).



Jérôme GIACOMINI
Consultant Infinite Square
Blog : http://blogs.infinitesquare.com/
Twitter : @jeromegiacomini

Les XamarinLib

Dans cet article j'ai essayé de regrouper les bibliothèques qui selon moi sont à connaître pour le développement d'applications via Xamarin.

Dans cet article nous verrons des bibliothèques permettant :

- D'accéder au système de fichiers avec une seule API (Windows/iOS/Android) ;
- De partager avec une seule API (Windows/iOS/Android) ;
- De connaître sa connectivité (Windows/iOS/Android) ;
- De lire et écrire dans les Settings (Windows/iOS/Android) ;
- D'afficher des graphiques (Windows/iOS/Android) ;
- D'afficher des images de façon optimisé (iOS/Android) ;
- De créer plus simplement des contraintes sur (iOS).

Accéder au système de fichier avec une seule API

PCLStorage *: il n'a jamais été aussi facile d'accéder au système de fichiers.*

Chaque système d'exploitation dispose d'un système de fichiers différent : on doit donc implémenter une manipulation du système de fichiers par plateforme.

Or **PCLStorage** permet d'accéder au système de fichiers avec une API unique.

- PCLStorage permet de manipuler le filesystem de manière uniforme ;
- Utilisable directement depuis une PCL ou .NET Standard ;
- Le code PCL ne contient pas directement les implémentations, qui sont « platform-specific » ;
- Entièrement asynchrone.
  Attention il ne semble plus maintenu, il va falloir trouver dans un futur proche une alternative.
  Si vous ciblez des plateformes compatibles avec .NET Standard 2.0 vous n'avez plus du tout besoin de cette lib.

Création d'un dossier et d'un fichier :

```
IFolder rootFolder = FileSystem.Current.LocalStorage;
IFolder folder = await rootFolder.CreateFolderAsync("MySubFolder",
CreationCollisionOption.OpenIfExists);
IFile file = await folder.CreateFileAsync("answer.txt",CreationCollisionOption.ReplaceExisting);
await file.WriteAllTextAsync("42");
```

Partager avec une seule API (iOS/Android)

PlugInShare est le moyen le plus simple de partager un message, un lien, ou de copier des informations dans le presse-papier.

Partage d'une url :

```
  public void ShareBlog()
{
  if(!CrossShare.IsSupported)
    return;

  CrossShare.Current.Share(new ShareMessage
  {
    Title = "Blog JGI",
    Text = "Le blog de JGI",
    Url = "http://jeromegiacomini.net/"
  });
}
```

Connaître sa connectivité

Xam.Plugin.Connectivity *: connaître la connectivé de son application simplement.*

Vérifier si l'application a accès à internet :

```
CrossConnectivity.Current.IsConnected
```

Vérifier si le device est connecté au Wifi :

```
var wifi  = ConnectionType.WiFi;

var connectionTypes = CrossConnectivity.Current.ConnectionTypes;
 if(!connectionTypes.Contains(wifi))
 {
  //You do not have wifi
  return null;
 }
```

Lire et écrire dans les Settings

Xam.Plugin.Settings *: mes paramètres sont multiplateformes.*

Permet de créer des settings d'applications de façon unifiée sur les 3 plateformes.

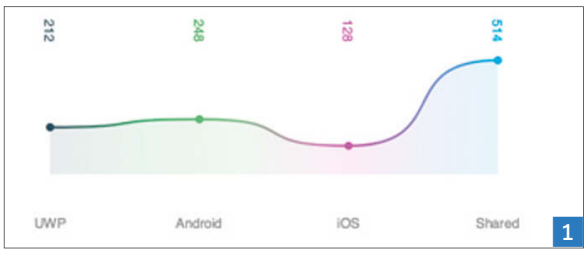
Créer un paramètre de type string :

```
private static ISettings AppSettings =>
  CrossSettings.Current;

public static string UserName
{
 get => AppSettings.GetValueOrDefault(nameof(UserName),string.Empty);
 set => AppSettings.AddOrUpdateValue(nameof(UserName), value);
}
```

Afficher des graphiques

Microcharts permet de créer des graphiques sur iOS et Android avec le même code.

Un contrôle Xamarin.Forms est aussi disponible. 1



1

Création d'un graphique :

```
var entries = new[]
{
  new Entry(200)
  {
    Label = "January",
    ValueLabel = "200",
  FillColor = SKColor.Parse("#266489")
  },
  new Entry(400)
  {
  Label = "February",
  ValueLabel = "400",
  FillColor = SKColor.Parse("#68B9C0")
  },
  new Entry(-100)
  {
  Label = "March",
  ValueLabel = "-100",
  FillColor = SKColor.Parse("#90D585")
  }
};

var chart = new LineChart() { Entries = entries };
```

Afficher des images de façon optimisée

1/ Sur Android

Square Picasso est la librairie *incontournable* pour afficher des images et les mettre cache sur Android.
Télécharge une image :

```
Picasso.With(context).Load("http://monimage.png") .Into(imageView);
```

2/ Sur iOS

SDWebImage est l'équivalent de Picasso mais sur iOS.
Il supporte notamment le format d'image GIF.
Comme Picasso, il télécharge les images en asynchrone et gère aussi le cache des images téléchargées.
Télécharge une image et affiche un placeholder durant le téléchargement :

```
using SDWebImage;
...

imageView.SetImage (
    url: new NSUrl ("http://www.monimage.png"),
    placeholder: UIImage.FromBundle ("placeholder.png")
  );
```

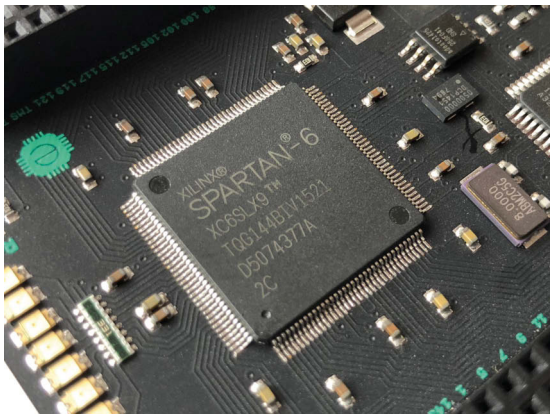
Créer plus simplement des contraintes

Cirrious.FluentLayout : permet de créer des contraintes iOS de façon plus simple.
Exemple de création de contraintes :

```
View.AddConstraints(
  fNameLabel.AtTopOf(View, vMargin),
  fNameLabel.AtLeftOf(View, hMargin),
  fNameLabel.ToLeftOf(sNameLabel, hMargin),

  sNameLabel.WithSameTop(fNameLabel),
  sNameLabel.AtRightOf(View, hMargin),
  sNameLabel.WithSameWidth(fNameLabel),

  fNameField.WithSameWidth(fNameLabel),
  fNameField.WithSameLeft(fNameLabel),
  fNameField.Below(fNameLabel, vMargin),

  sNameField.WithSameLeft(sNameLabel),
  sNameField.WithSameWidth(sNameLabel),
  sNameField.WithSameTop(fNameField));
```

Avec toutes ces bibliothèques vous êtes prêt à développer des apps Xamarin en un rien de temps.
Happy coding :)

Arnaud Boudou
développeur chez ekito
co-fondateur du Fab Lab Sud31



À la découverte du FPGA

En tant que Maker ou simplement hobbyiste passionné, vous vous êtes peut-être lancé dans le développement autour des microcontrôleurs (Arduino, MSP43x, PIC, etc.). Et un jour, vous avez entendu parler des FPGA, sans forcément savoir de quoi il s'agissait exactement, à part peut-être de quelque chose comme les microcontrôleurs, mais en plus compliqué (mystérieux ?). Cet article est là pour vous aider à aborder ce nouveau continent.

Avant d'aborder plus en détails le FPGA, rappelons de quoi est généralement constitué un microcontrôleur :

- un processeur, qui exécute des instructions ;
- de la RAM, qui stockera les données temporaires liées à l'exécution du programme ;
- de la ROM, qui stockera le programme à destination du processeur ;
- des périphériques (ports séries, entrées / sorties numériques et / ou analogiques, etc.) pour communiquer avec le monde extérieur.

Toutes ces parties sont du point de vue physique des ensembles de composants (transistors, condensateurs, résistances…) gravés dans le silicone, à défaut de marbre, et définitivement figés. Pour ajouter de nouvelles fonctionnalités, vous devrez concevoir un nouveau circuit.

Le FPGA, qu'est-ce que c'est ?

À côté, nous avons le FPGA, qui pourrait être vu comme une feuille blanche. Sur cette feuille blanche, vous allez pouvoir dessiner à peu près ce que vous voulez. Un FPGA peut donc aussi être vu comme un composant électronique qui ne sait rien faire une fois sorti de sa boîte. Rentrons maintenant dans les détails. FPGA est un acronyme signifiant Field-Programmable Gate Array (soit en français : réseau de portes programmables in situ). C'est un circuit programmable (« programmable » devant être compris comme « configurable ») composé d'une grille d'éléments logiques simples. Programmer un FPGA signifie donc interconnecter ces éléments logiques pour y implémenter la fonctionnalité voulue. Cette fonctionnalité peut être : un additionneur, un clignoteur de LED, un circuit logique ou même un processeur. Et comme les éléments logiques de base ne sont normalement pas connectés entre eux, il est même possible d'implémenter plusieurs fonctionnalités indépendantes les unes des autres. Vous êtes seulement limité par vos compétences et le nombre d'éléments logiques de votre FPGA.

J'ai parlé juste avant d'éléments logiques. Un élément logique est en général composé :

- d'une table de correspondance (LUT – Look Up Table) à plusieurs entrées et une sortie, servant à implémenter une équation en logique booléenne;
- une bascule (flip-flop) permettant de stocker l'état de la sortie de la table de correspondance.

Les éléments logiques sont reliés entre eux par une matrice de routage. Outre le circuit de programmation de la matrice et des éléments logiques, un FPGA peut contenir des circuits spécialisés, comme des multiplicateurs (coûteux à implémenter en logique booléenne), ou des blocs de RAM. L'intégration de ces circuits permet de laisser plus d'éléments logiques à votre disposition.

Les éléments logiques et la matrice de routage étant constitués de cellules SRAM, cela signifie qu'en cas de coupure de l'alimentation, le programme sera perdu. Il est donc généralement associé à une puce FPGA, un circuit externe contenant en ROM la configuration FPGA qui sera copiée sur la puce lors de l'allumage de l'ensemble.

Le FPGA, quelle utilité ?

Maintenant que nous avons une idée plus claire de ce que sont les FPGA, intéressons-nous à leur utilité.

Tout d'abord, les FPGA sont utilisés comme systèmes de prototypage de circuits avant que ceux-ci soient fabriqués de manière industrielle pour commercialisation. Faire fabriquer des circuits en dur coûtant plusieurs milliers d'euros / dollars / autre devise, toute erreur lors des tests de conception peut rapidement tourner au cauchemar financier. Passer par l'étape FPGA, qui est librement reconfigurable, permet de valider la logique via de simples reprogrammations, même si les performances sont inférieures au circuit final. Une autre utilité des FPGA est de pouvoir obtenir des circuits conçus pour des domaines spécifiques. Plutôt que d'utiliser un processeur exécutant un programme implémentant la solution souhaitée, il est possible d'implémenter cette même fonctionnalité dans un circuit spécialisé qui sera plus performant qu'un processeur généraliste. Par exemple, il est possible d'implémenter des algorithmes de reconnaissance d'images sur des FPGA plutôt que de faire mouliner un processeur généraliste.

Dernière utilisation d'un FPGA à laquelle je pense, celle de composant « généraliste » au sein d'équipements électroniques grand public. J'ai ainsi croisé une puce FPGA en désossant une imprimante laser. L'utilisation d'une puce FPGA permet d'apporter à la machine des fonctionnalités basiques qui pourront être améliorées et / ou corrigées par simple reconfiguration de la puce. Dans ce cas, la nouvelle configuration peut être apportée par une mise à jour des pilotes ou du firmware de l'équipement.

En reprenant chacun des points ci-dessous, il est possible de faire le parallèle avec les solutions de minage du bitcoin.

Au début étaient les programmes de minage se servant du **CPU généraliste** d'un ordinateur. Puis, quand la difficulté est devenue trop importante, les mineurs sont passés minage sur **GPU** ; des processeurs moins généralistes que les CPU et spécialisés dans les traitements mathématiques et parallèles.

Puis, quand la difficulté est devenue trop importante, les mineurs sont passés au circuit FPGA spécialisé. Le circuit était donc dédié uniquement au minage de bitcoin avec la logique ad-hoc, et chaque fois que des améliorations dans les algorithmes étaient disponibles, le circuit était reconfiguré avec une nouvelle version du programme.

Puis, quand la difficulté est devenue trop importante, les mineurs sont passés au **circuit ASIC** (Application-specific integrated circuit - circuit intégré propre à une application) : après le prototype FPGA, la puce est gravée en dur et suivant la même logique programmée. Les circuits ASIC offrant des performances nettement plus importantes que les FPGA, le minage de bitcoin a pu se poursuivre en toute quiétude.

Développer sur FPGA, les solutions

Maintenant que vous avez décidé de vous lancer dans le développement sur FPGA, vous allez devoir faire plusieurs choix, les deux principaux étant :

- quelle marque de FPGA ?
- quel langage ?

Les marques les plus fréquemment rencontrées sont les suivantes : Xilinx, Intel (anciennement Altera) et Lattice. Chacune de ces marques proposera à la fois ses gammes de FPGA, avec leurs environnements de développement dédiés. Pour la suite, je ne vais pas trop m'étendre sur les différences entre chaque gamme, cela nous ferait sortir du cadre de cet article. 1

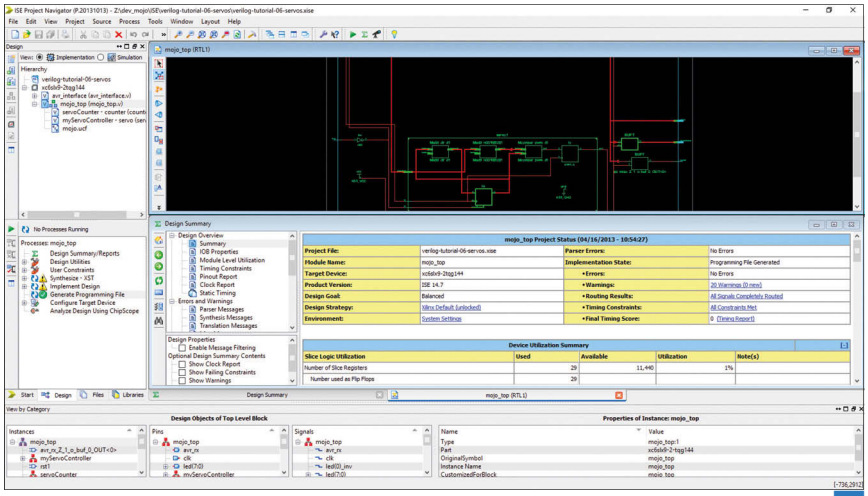
Xilinx

Cette marque propose principalement comme gamme de FPGA deux grandes familles : Spartan, et Virtex. En fonction de la famille visée, l'environnement de développement à utiliser n'est pas le même.

- ISE Design Suite (Windows / Linux) : cet environnement de développement cible la gamme Spartan-6 et Virtex-6 (ou versions précédentes). Il existe une version gratuite d'ISE, nommée « ISE Design Suite : WebPACK Edition », qui est plus limitée que la version normale, principalement sur des fonctions avancées. Gros défaut d'ISE, c'est que cette suite logicielle n'est plus maintenue par Xilinx, fonctionne mal sous Windows 10 (sauf bricolages pour l'installation) et pas du tout avec les distributions Linux utilisant le noyau 4.x.
- Vivado Design Suite (Windows / Linux) : cet environnement est cible la gamme Spartan-7 et Virtex-7. Là aussi, il existe une version gratuite nommée « Vivado HL WebPACK », elle aussi limitée en fonctionnalités. Contrairement à ISE, cet environnement de développement fonctionne parfaitement avec les versions récentes de Windows et Linux.

Intel

Plus précisément, Intel a racheté Altera en 2015, et a conservé son activité dans le domaine FPGA. La gamme de FPGA que vous êtes susceptible de rencontrer le plus fréquemment est la famille Cyclone. L'environnement de développement utilisé pour les produits Intel est Quartus (Windows / Linux). De la même manière que pour Xilinx, il existe une version gratuite limitée en fonctionnalités et en puces supportées. Mais elle suffira amplement pour se lancer dans le développement FPGA.



1

Lattice

Lattice est principalement connu chez les hobbyistes pour sa gamme de FPGA à faible coût, nommée iCE. L'environnement de développement indiqué en est Lattice Diamond (Windows / Linux). Il est possible d'obtenir une licence gratuite, qui sera limitée en produits supportés.

Il est à noter qu'un projet de développement d'une chaîne de compilation libre pour la gamme iCE40 existe. Cette chaîne de compilation vise à supporter Windows, Linux mais aussi macOS.

Vous noterez à la lecture des paragraphe précédents que, si vous préférez travailler sous macOS, comme c'est mon cas, vous allez devoir dégainer une machine virtuelle pour vous lancer dans le développement FPGA.

Une fois votre famille de prédilection choisie, il vous faut maintenant trouver une carte de développement. Pour vos débuts, pas besoin de vider le compte en banque, vous pouvez trouver le nécessaire pour une centaine d'euros. Honnêtement, il est assez difficile de conseiller une carte en particulier. Une liste plutôt exhaustive est disponible ici : https://joelw.id.au/FPGA/CheapFPGADevelopmentBoards.

Si vous souhaitez utiliser des FPGA Xilinx, j'aurais quand même tendance à vous déconseiller les cartes à base de Spartan-6 ou Spartan-3, qui vous obligeront à utiliser ISE (sauf bien entendu si vous avez la possibilité d'avoir un ordinateur un peu ancien sous le coude). Autre point à prendre en compte : la connexion à l'ordinateur de développement. Certaines cartes fournissent un connecteur USB qui sera directement reconnu par l'environnement de développement, d'autres auront un connecteur USB qui nécessitera un utilitaire dédié (fourni par le fabricant de la carte) pour le transfert du programme. Enfin certaines cartes ne fourniront qu'un connecteur JTAG, vous laissant la responsabilité de trouver le programmeur adéquat. Vous avez maintenant choisi votre famille de FPGA, votre carte de développement. Il ne vous reste plus qu'à

choisir le langage que vous allez utiliser. Les langages utilisés pour le développement FPGA sont appelés « HDL », pour « Hardware Description Langage ». En effet, quand on développe pour du FPGA, on n'écrit pas une liste d'instructions qui vont se dérouler les unes après les autres. À la place, on écrit la description d'un circuit au niveau logique. Cette description sera ensuite synthétisée en une série de portes logiques. Toutes les « instructions » s'exécuteront en parallèle et seront synchronisées sur la fréquence d'horloge du circuit.

Les deux langages majoritairement utilisés sont Verilog et VHDL. Le choix d'un langage est une simple question de préférence personnelle, ou même du langage utilisé pour les tutoriaux de votre carte de développement. Pour la petite histoire, VHDL est principalement utilisé en Europe, et Verilog de l'autre côté de l'Atlantique.

Tarifs

- Xilinx Vivado : de 2995 $ à 4295 $ en fonction de l'édition et du type de licence (fixe ou flottante)
- Xilinx ISE : une licence Vivado donne aussi droit à une licence ISE. Il n'est pas possible d'obtenir de licence ISE séparément.
- Intel Quartus : de 2995 $ à 4995 $ en fonction de l'édition et du type de licence

Premier programme sur FPGA

Il est maintenant temps d'écrire vos premiers programmes pour FPGA. Les exemples donnés sont pour la carte de développement Mojo v3 (https://embeddedmicro.com/products/mojo-v3.html), équipée d'une puce Spartan-6, et donc nécessitant Xilinx ISE, et seront écrits en Verilog. Ces exemples de code devraient être relativement aisés à adapter à une autre carte de développement. À cette puce Spartan est associé un microcontrôleur AVR faisant le lien avec l'ordinateur pour la programmation de la puce FPGA, et stockant le programme afin de pouvoir le réinstaller en cas de coupure de l'alimentation. Je partirai du principe que le projet que vous initialiserez dans votre environnement de développement sera correctement configuré pour votre carte. Il y a trop de combinaisons possibles pour que j'aborde ce sujet, je vous conseille donc de vous tourner vers la documentation de votre constructeur.

Vous le savez sûrement, mais le « hello world » du développement mettant en jeu de l'électronique est le clignoteur de LED : le résultat est facilement visible, et facilement modifiable. Je ne vais donc pas déroger à la tradition. En revanche, je vais commencer par le clignoteur de LED manuel…

Le clignoteur de LED manuel

La carte Mojo V3 possède entre autres 8 LEDs côte à côte et un bouton reset. Le but du jeu est de faire s'allumer une des LEDs lors de l'appui sur le bouton.

Lors de la création de votre projet, un fichier « mojo_top.v » est créé. Celui-ci représente le module dit « top level », qui implémentera d'éventuels sous-modules, et représentera l'état final du circuit. Un sous-module représente quant à lui une fonctionnalité de votre circuit (un compteur, un additionneur, etc.).

L'extension « .v » correspond à un fichier de langage Verilog. Chaque éventuel sous-module doit être dans son propre fichier pour des questions de lisibilité et maintenance.

Un module commence par la ligne :

```
module nom_module(
 les_parametres_du_module
);
```

Et se termine par la ligne :

```
endmodule
```

Notre module « mojo_top » commence par les lignes suivantes :

```
module mojo_top(
  // Entrée horloge à 50 MHz
  input clk,
  // Entrée reliée au bouton reset (active à l'état bas)
  input rst_n,
  // Entrée venant de l'AVR de programmation, état haut quand AVR prêt
  input cclk,
  // Sortie vers les 8 LEDs embarquées
  output[7:0]led,
  // Connexions vers le port SPI de l'AVR
  output spi_miso,
  input spi_ss,
  input spi_mosi,
  input spi_sck,
  // Sélection du convertisseur analogique vers numérique de l'AVR
  output [3:0] spi_channel,
  // Connexions séries vers l'AVR
  input avr_tx, // AVR Tx => FPGA Rx
  output avr_rx, // AVR Rx => FPGA Tx
  input avr_rx_busy // AVR Rx buffer full
 );
```

Cette déclaration indique l'ensemble des signaux qui entrent et qui sortent du module. Vous remarquerez pour l'entrée « reset » un suffixe « _n ». C'est une convention de nommage indiquant que cette entrée est active à l'état bas.

La sortie « led » est sous la forme « output[7:0] ». Cela signifie qu'il s'agit de 8 connexions en sortie, indexées de 7 à 0. Rien n'empêche d'écrire « output[10:3] » ou « output[8:1] », ou même « output[0:7] », mais le plus sûr est de coller à la convention, à moins d'avoir de très bonnes raisons de s'en écarter.

Après la déclaration du module, nous allons ajouter une connexion venant du bouton reset, mais inversant sa valeur. Ce qui signifie qu'elle sera active à l'état haut.

Il faut donc rajouter la ligne suivante :

```
wire rst = ~rst_n;
```

Le tilde devant « rst_n » signifie qu'on inverse sa valeur.

Les quatre lignes suivantes du projet sont :

```
assign spi_miso = 1'bz;
assign avr_rx = 1'bz;
assign spi_channel = 4'bzzzz;

assign led = 8'b0;
```

Le mot clé « assign » permet de donner une valeur à une connexion

précédemment déclarée. La valeur est sous la forme suivante : la largeur en bit de la valeur, une apostrophe de séparation, le format de la valeur (« b » pour bit, « h » pour hexadécimal, « d » pour décimal), et la valeur proprement dite.
Dans le cas des valeurs en bits, on a quatre-sous valeurs possibles : « 0 », « 1 » (jusque-là, normal), « z » (qui signifie état de haute impédance, ou déconnecté) et « x » (qui signifie valeur inconnue, ou on ne s'en préoccupe pas).
Du coup, on peut lire les choses suivantes :

- « spi_miso » se voit affecter une valeur de 1 bit, égale à « z » ;
- « avr_rx » de même ;
- « spi_channel » se voit affecter une valeur de 4 bits, égale à « zzzz » ;
- « led » se voit affecter une valeur de 8 bits, égale à « 0 » (ou plus précisément « 00000000 »).

Au final, les trois premières connexions sont déconnectées par sécurité (elles relient l'AVR de programmation à la puce FPGA), et les 8 LEDs sont éteintes.
Nous allons maintenant remplacer la dernière ligne par les deux lignes suivantes :

```
assign led[6:0] = 7'b0;
assign led[7] = rst;
```

Dans ce cas, on forcera les LEDs 0 à 6 à être éteintes (valeur 0), et la LED 7 sera branchée sur le signal inversé venant du bouton reset. On termine enfin le module par la ligne

```
endmodule
```

Le code complet (nettoyé des commentaires) est donc le suivant :

```
module mojo_top(
  input clk,
  input rst_n,
  input cclk,
  output[7:0]led,
  output spi_miso,
  input spi_ss,
  input spi_mosi,
  input spi_sck,
  output [3:0] spi_channel,
  input avr_tx,
  output avr_rx,
  input avr_rx_busy
 );

 wire rst = ~rst_n;

 assign spi_miso = 1'bz;
 assign avr_rx = 1'bz;
 assign spi_channel = 4'bzzzz;

 assign led[6:0] = 7'b0;
 assign led[7] = rst;

endmodule
```

Une fois le programme envoyé sur la carte, vous pourrez allumer la LED 7 en appuyant sur le bouton reset, et l'éteindre en relâchant le bouton. Dans ce programme, il n'y a pas de notion de boucle ou autre, nous avons simplement créé des connexions entre le bouton reset et la LED 7, avec un inverseur de signal entre les deux. Vous pouvez voir sur l'image ci-dessous le résultat de ce qui va être implémenté sur la puce FPGA. De gauche à droite, vous avez le signal d'entrée « rst_n », le passage par l'inverseur, et la sortie vers les LEDs (le détail sur la LED exactement branchée à l'inverseur n'apparaît pas). 2
Appuyer sur un bouton pour faire clignoter une LED, c'est fatigant. Il serait donc bien de rendre automatique ce clignotement. Nous allons donc passer au clignoteur automatique.

Clignoteur automatique

Pour cela, nous allons créer un nouveau sous-module, nommé « blinker », dans un fichier justement nommé… « blinker.v » (bravo à ceux qui n'ont pas encore décroché).
L'entête du module va prendre la forme suivante :

```
module blinker(
  input clock,
  input reset,
  output blink
 );
```

Ce module aura en entrée le signal d'horloge (celle qui anime notre puce, ici cadencée à 50 MHz), le signal du bouton reset, et aura comme sortie l'état du clignotement (éteint / allumé)
Nous allons déclarer dans ce module deux registres de 25 bits.

```
reg [24:0] counter_d, counter_q;
```

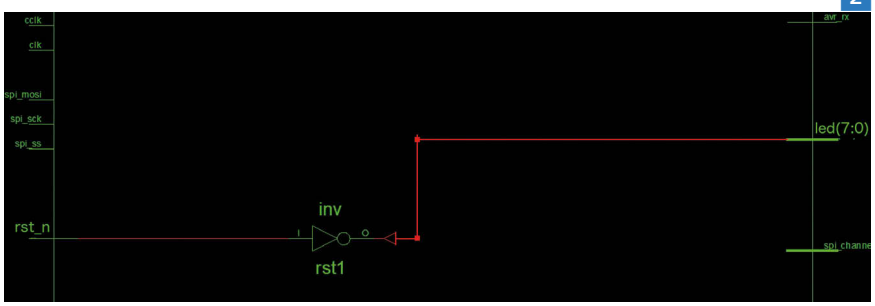
Ces registres seront implémentés sous la forme d'une chaine de 25 bascules de type D. Une bascule de type D est une mémoire de 1 bit avec une entrée D et une sortie Q. À chaque front d'horloge montant, la bascule va copier sur Q le signal présent sur D. Entre deux fronts montants d'horloge, la valeur de Q est conservée.
Nous décidons ensuite que la sortie « blink » du module sera reliée au 25ème bit du registre « counter_q ».

```
assign blink = counter_q[24];
```

À chaque fois que « counter_q » sera modifié, « counter_d » devra recevoir la valeur de « counter_q » incrémentée de 1 bit.

```
always @(counter_q) begin
  counter_d = counter_q + 1'b1;
end
```

2



À chaque front d'horloge, nous allons copier la valeur de « counter_d » dans « counter_q », déclenchant le bloc précédent. De plus, tant que le bouton reset sera maintenu appuyé, le compteur « counter_q » sera remis à 0.

```
always @(posedge clock) begin
 if (reset) begin
  counter_q <= 25'b0;
 end else begin
  counter_q <= counter_d;
 end
end
```

Et nous terminons notre sous-module par la ligne suivante :

```
endmodule
```

Le code complet du module est donc le suivant :

```
module blinker(
  input clock,
  input reset,
  output blink
 );

 reg [24:0] counter_d, counter_q;

 assign blink = counter_q[24];

 always @(counter_q) begin
  counter_d = counter_q + 1'b1;
 end

 always @(posedge clock) begin
  if (reset) begin
   counter_q <= 25'b0;
  end else begin
   counter_q <= counter_d;
  end
 end

endmodule
```

Vous aurez peut-être remarqué que l'assignation de valeur de « counter_q » se fait avec l'opérateur « <= » et non « = ». Cela signifie que cette assignation est non bloquante. Le résultat est que toutes les valeurs assignées avec cet opérateur au sein d'un même bloc sont assignées en parallèle et non pas séquentiellement.
Grosso modo, ce compteur va incrémenter un registre de 25 bits à chaque cycle d'horloge. Pendant la première moitié du temps, le bit 24 (le plus à gauche) sera à zéro (donc des valeurs 000000000000000000000000 à 011111111111111111111111), la sortie blink en fera de même. Et pendant l'autre moitié du temps, (donc des valeurs 100000000000000000000000 à 111111111111111111111111), le bit 24 sera à un, et de même pour la sortie blink. On aura donc 16 777 216 cycles d'horloge à 0 et autant à 1, ce qui fait pour une horloge à 50 MHz environ 0,34s à 1 et 0,34s à 0.
Il nous faut maintenant modifier le code du module « mojo_top.v » de la manière suivante. Il nous faut remplacer les lignes :

```
assign led[6:0] = 7'b0;
assign led[7] = rst;
```

Par les lignes suivantes :

```
assign led[7:1] = 7'b0;

blinker awesome_blinker (
 .clock(clk),
 .reset(rst),
 .blink(led[0])
);
```

Ces lignes de code implémentent un module « blinker » sous le nom « awesome_blinker », branchent les entrées du sous-module (ce qui commence par un point) avec les lignes d'horloge et de reset du module principal, et branchent la sortie du sous-module à la LED numéro 0.
Le code complet du module « mojo_top » est donc le suivant :

```
module mojo_top(
  input clk,
  input rst_n,
  input cclk,
  output[7:0]led,
  output spi_miso,
  input spi_ss,
  input spi_mosi,
  input spi_sck,
  output [3:0] spi_channel,
  input avr_tx,
  output avr_rx,
  input avr_rx_busy
 );

 wire rst = ~rst_n;

 assign spi_miso = 1'bz;
 assign avr_rx = 1'bz;
 assign spi_channel = 4'bzzzz;

 assign led[7:1] = 7'b0;

 blinker awesome_blinker (
  .clock(clk),
  .reset(rst),
  .blink(led[0])
 );

endmodule
```

A vous de coder !

Voilà, ce dernier bout de code conclut cet article sur la découverte de la programmation pour FPGA. J'espère que cet aperçu vous a donné envie d'aller plus loin dans ce domaine, qui sort complètement de l'ordinaire pour un développeur logiciel. •

Sur abonnement ou en kiosque

Le magazine des pros de l'IT

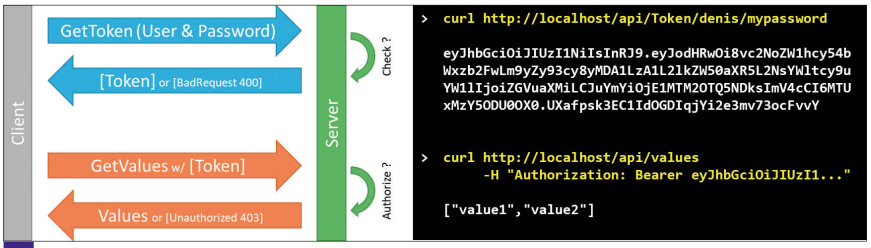
Mais aussi sur le web

L'INFORMATICIEN
www.linformaticien.com

Denis Voituron
MVP, Speaker & Podcaster DevApps.be
.NET Senior Architect
@ http://oniryx.be
@DenisVoituron

JSON Web Token, sécuriser une WebAPI grâce au jeton JWT

JSON Web Token (JWT) est un standard ouvert (RFC 7519) pour échanger de l'information de manière sécurisée via un jeton signé. Par exemple un serveur pourrait émettre un jeton possédant l'affirmation "utilisateur identifié en tant qu'administrateur" et le fournir au client. Le client pourrait alors vérifier le jeton pour prouver que l'utilisateur est identifié en tant qu'administrateur (Wikipedia).



1

Personnellement, c'est une solution que je trouve élégante car :

- Elle est simple à mettre en place : **quelques lignes de configuration** et l'utilisation de l'attribut [**Authorize**] ;
- Elle me permet de gérer facilement **mes propres règles de sécurité** lors de la vérification du nom d'utilisateur et du mot de passe ;
- Et elle est surtout très facile à tester dans les applications clientes : il suffit d'ajouter un en-tête HTTP "**Authorization: Bearer [Token]**".

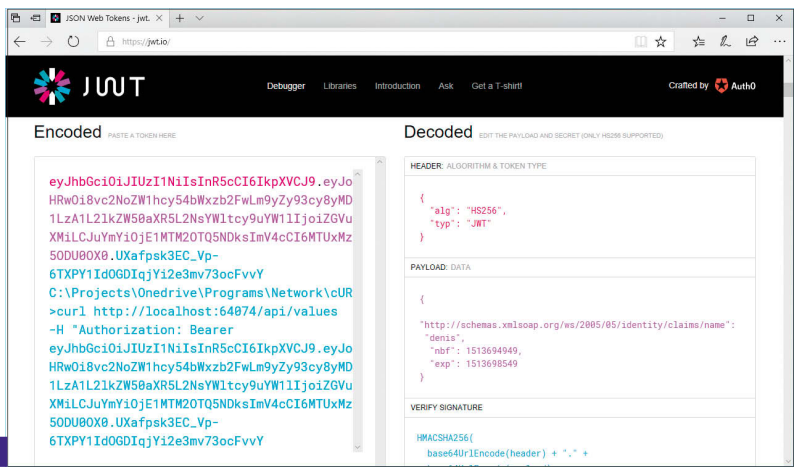
Flux général

Le principe est d'appeler une méthode qui génère un jeton (token) JWT contenant notamment, une date d'expiration et quelques métadonnées, signé pour éviter les altérations. Ce jeton est ensuite envoyé à toutes les requêtes, via l'en-tête HTTP. 1

Lorsque vous avez généré un jeton JWT, vous pouvez facilement le valider depuis le site web http://jwt.io.

Contenu du jeton JWT

Un jeton JWT est composé de trois parties : un en-tête, une charge utile et une signature :



2

- **L'en-tête** indique quel algorithme a été utilisé pour générer la signature (par exemple HMAC-SHA256) ;
- **Le Payload** est variable en fonction de l'application, mais il est indiqué dans les spécifications de JWT d'inclure une empreinte temporelle à la création : nbf (not before – date et heure d'utilisation) et exp (date et heure d'expiration) ;
- **La signature** est obtenue via l'algorithme spécifié dans l'en-tête appliqué au jeton et à la clé. Elle permet de garantir que le Token n'a pas été modifié depuis sa création. 2

Configuration des WebAPI

La première étape consiste à créer un projet **ASP.NET Core**, via Visual Studio 2017 ou via Visual Studio Code. Ce projet pourra héberger vos WebAPI.

- Démarrez **Visual Studio 2017 (1)** et créez un projet depuis les menus File / New / Project / Visual C# / .Net Core / ASP.NET Core Web Application.
- Ajoutez les librairies NuGet **System.IdentityModel.Tokens.Jwt** et **System.Runtime.Serialization.Json**, via le menu Project / Manage NuGet Packages.
- Adaptez la classe **Startup.cs** pour configurer le service JWT : ajoutez et configurez l'authentification JWT comme ci-dessous.

```
public class Startup
{
  public void ConfigureServices(IServiceCollection services)
  {
    services.AddMvc();

    // Configure the JWT Authentication Service
    services.AddAuthentication(options =>
    {
      options.DefaultAuthenticateScheme = "JwtBearer";
      options.DefaultChallengeScheme = "JwtBearer";
    })
    .AddJwtBearer("JwtBearer", jwtOptions =>
    {
      jwtOptions.TokenValidationParameters = new TokenValidationParameters()
      {
        // The SigningKey is defined in the TokenController class
        IssuerSigningKey = TokenController.SIGNING_KEY,
```

(1) Visual Studio Community est téléchargeable gratuitement depuis https://www.visualstudio.com/fr/vs/community

```
            ValidateIssuer = false,
            ValidateAudience = false,
            ValidateIssuerSigningKey = true,
            ValidateLifetime = true,
            ClockSkew = TimeSpan.FromMinutes(5)
        };
    });
  }

  public void Configure(IApplicationBuilder app, IHostingEnvironment env)
  {
    app.UseAuthentication();  // Must be before app.UseMvc
    app.UseMvc();
  }
}
```

- Créez la classe **TokenController.cs** et ajoutez la méthode de génération du Token. L'algorithme utilisé est basé sur SHA256. Il demande une clé secrète d'au moins 128 bits (16 caractères). Le site **https://randomkeygen.com** vous en donnera une rapidement.

```
public class TokenController : Controller
{
  private const string SECRET_KEY = "TQvgjeABMPOwCycOqah5EQu5yyVjpmVG";
  public static readonly SymmetricSecurityKey SIGNING_KEY = new
          SymmetricSecurityKey(Encoding.UTF8.GetBytes(SECRET_KEY));

  [HttpGet]
  [Route("api/Token/{username}/{password}")]
  public IActionResult Get(string username, string password)
  {
    if (username == "denis") // TODO: Validate your username and password HERE.
      return new ObjectResult(GenerateToken(username));
    else
      return BadRequest();
  }

  // Generate a Token with expiration date and Claim meta-data.
  // And sign the token with the SIGNING_KEY
  private string GenerateToken(string username)
  {
    var token = new JwtSecurityToken(
      claims:   new Claim[] { new Claim(ClaimTypes.Name, username) },
      notBefore: new DateTimeOffset(DateTime.Now).DateTime,
      expires:  new DateTimeOffset(DateTime.Now.AddMinutes(60)).DateTime,
      signingCredentials: new SigningCredentials(SIGNING_KEY,
                          SecurityAlgorithms.HmacSha256)
      );

    return new JwtSecurityTokenHandler().WriteToken(token);
  }
}
```

Le jeton ainsi généré contient le nom de l'utilisateur (vous pouvez facilement ajouter un Claim de type Role), la plage de temps pendant laquelle le Token est valide, et l'algorithme de signature utilisé.

- La dernière étape consiste à demander une authentification des méthodes à sécuriser. Pour cela, utilisez l'attribut **[Authorize]**. Lors de la génération du Token (étape précédente), si vous aviez défini un *Claim(ClaimTypes.Role, "Admin")*, vous auriez pu l'utiliser dans cet attribut *[Authorize(Roles = "Admin")]*.

```
[Route("api/[controller]")]
public class ValuesController : Controller
{
  // GET api/values
  [HttpGet]
  [Authorize]
  public IEnumerable<string> Get()
  {
    return new string[] { "value1", "value2" };
  }
```

Vérification de l'autorisation des méthodes

En exécutant votre projet WebAPI, vous ne pouvez plus appeler l'action api/values (définie dans la classe ValuesController), car vous recevrez une erreur de type "**401. Unauthorized**". Pour pouvoir exécuter cette action, vous devez préalablement obtenir un Token et l'utiliser dans l'en-tête de la requête.

Pour tester vos requêtes, vous pouvez utiliser Postman ou la commande CURL. Exemple.

```
> curl -i http://localhost/api/values

HTTP/1.1 401 Unauthorized
WWW-Authenticate: Bearer
X-Powered-By: ASP.NET
Content-Length: 0
```

```
> curl -i http://localhost/api/token/denis/mypassword

HTTP/1.1 200 OK
X-Powered-By: ASP.NET

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJodHRwOi8vc2NoZW1hcy54bWxzb2FwLm9yZy93
cy8yMDA1LzA1L2IkZW50aXR5L2NsYWltcy9uYW1lIjoiZGVuaXMiLCJuYmYiOjE1MTM3MDM5
OTIsImV4cCI6MTUxMzcwNzU5Mn0.LepDs7Rm6bvcANUDHI2QaKW0A8ooN5iuALI7k9400HU
```

```
> curl -i http://localhost/api/values -H "Authorization: Bearer eyJhbGciOi..."

HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
X-Powered-By: ASP.NET

["value1","value2"]
```

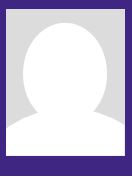
Conclusion

JSON Web Token (JWT) est un standard ouvert qui ne demande que quelques lignes de configuration et qui permet de sécuriser facilement les WebAPI développées en C#. Ce protocole étant normalisé, il est compatible avec d'autres plateformes et d'autres langages : Java, Python, JavaScript, Go, Ruby, …

Le jeton utilisé lors des échanges de données est signé et peut contenir vos propres métadonnées. Il est également très facile d'intégrer les WebAPI dans des projets de tests unitaires. •

Exemple en vidéo : **https://youtu.be/3I4KHG2mppc**

Code source : **https://github.com/dvoituron/SampleJWT**



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

Réalisez un clone du célèbre jeu Flappy Bird sous Android

Il y a de cela 4 ans, la folie Flappy Bird s'emparait de l'univers des applications mobiles. Un simple jeu 2D réalisé par un développeur indépendant atteignait la première place des classements de l'App Store d'Apple et du Play Store de Google. Ce succès aura démontré qu'un jeu simple bien conçu au fort pouvoir addictif pouvait connaître un immense succès. Dans cet article, nous vous proposons de créer un clone de Flappy Bird sous Android afin de découvrir comment créer un jeu 2D simple.

La simplicité peut quelquefois être gage de succès. Cela l'est encore plus lorsque cette simplicité est couplée à un fort pouvoir addictif. Car c'est bien ce côté addictif qui est à l'origine du succès phénoménal qu'aura connu Flappy Bird au cours de l'année 2014. Le gameplay de Flappy Bird est simple et repose avant tout sur l'agilité du joueur. Ce dernier doit faire avancer un oiseau dans un décor à défilement horizontal en tapotant sur l'écran tactile de son appareil mobile.

La principale difficulté du jeu étant que le joueur doit éviter des obstacles en forme de tuyaux présents en haut et en bas de l'écran. A chaque passage de tuyau, le joueur marque un point. Tout au long de sa progression dans l'environnement, le joueur peut collecter des pièces afin de marquer des points supplémentaires. La partie ne s'arrête que lorsque le joueur touche un tuyau ou heurte le sol représenté par le bas de l'écran. Le succès de Flappy Bird tient également dans ses graphismes rappelant ceux de Super Mario World notamment. Le jeu devient rapidement addictif voir frustrant lorsque l'on commence à tenter de se mesurer aux autres joueurs autour du monde.

De par sa simplicité, le jeu Flappy Bird se révèle un parfait cas d'étude pour apprendre à créer un premier jeu 2D simple sous Android. Dans le cadre de cet article, nous allons donc réaliser un clone de Flappy Bird sans aucune bibliothèque externe pour que vous puissiez réellement découvrir les bases de la création d'un jeu 2D. Notre clone aura pour nom Flying Bird.

Préparation des sprites et des sons

La première étape dans la création de notre jeu Flying Bird va consister à récupérer les éléments graphiques qui se cachent en partie derrière le succès de Flappy Bird. Dans le monde de la programmation de jeux, ces éléments graphiques sont appelés Sprites. Tous nos Sprites seront placés au sein du dossier /res/drawable dans l'arborescence de notre projet d'application Android. La figure 1 permet d'avoir un aperçu de tous les Sprites qui seront utilisés par notre jeu Flying Bird. 1

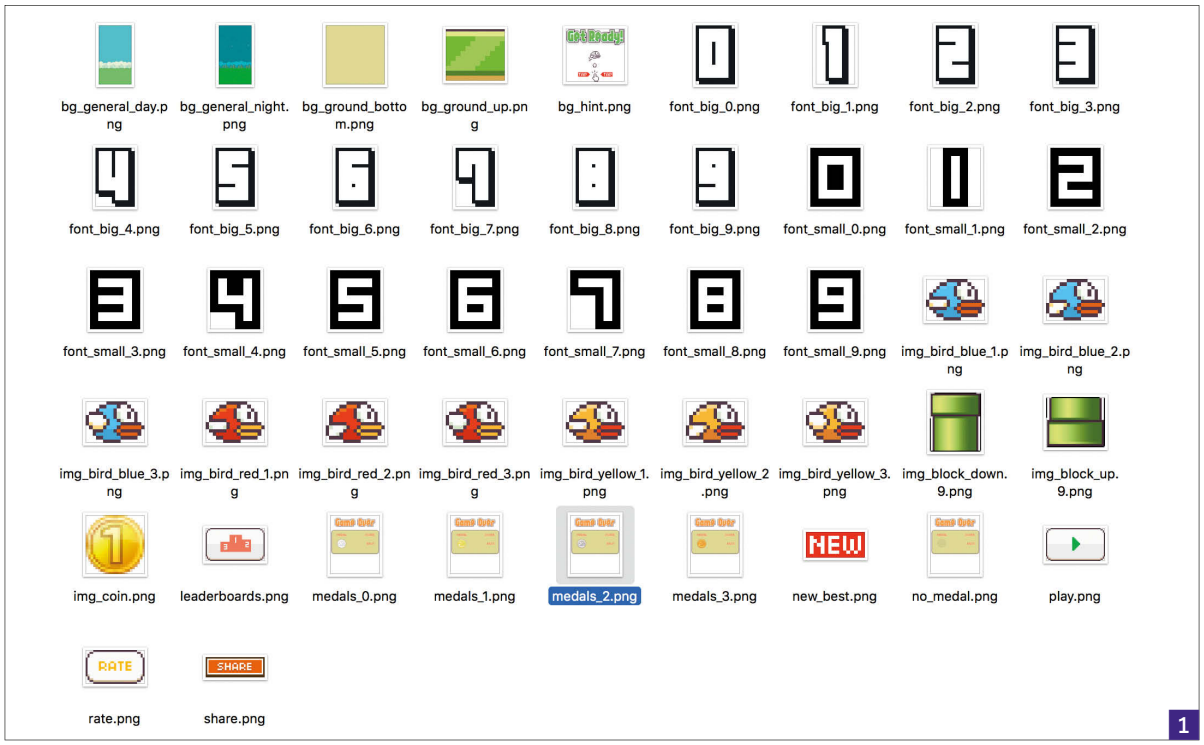
Comme vous pouvez le constater, on y retrouve : l'environnement du jeu en mode jour et en mode nuit, le sol, le Sprite qui donnera avec les tuyaux l'impression de défilement horizontal, les chiffres utilisés pour l'affichage du score, les différents Sprites de l'oiseau, les tuyaux, et enfin les Sprites utilisés en fin de partie par exemple pour afficher les récompenses. Notre clone ne saurait être réaliste sans la reproduction de l'univers sonore du jeu initial. Pour ce faire, nous plaçons les différents sons du jeu au sein du dossier assets. Parmi les différents sons que nous utiliserons, on retrouvera le son émis lorsque le joueur marque un point ou que l'oiseau meurt.

Création de l'interface utilisateur

L'interface utilisateur sera principalement constituée d'une zone de dessin sur laquelle nous rendrons les différents éléments dynamiques de notre jeu. Les éléments statiques peuvent quant à eux être définis au sein du layout de notre activité principale. Le décor de Flying Bird est l'élément statique que nous allons définir au sein d'une *ImageView* prenant l'ensemble de l'écran. La partie dynamique sera quant à elle dessinée au fur et à mesure de l'évolution du jeu au sein d'un composant *SurfaceView*. Ceci nous donne le layout suivant pour notre jeu :

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:id="@+id/main"
  android:layout_width="match_parent"
  android:layout_height="match_parent">

  <ImageView
    android:id="@+id/iv_background"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:scaleType="centerCrop"
    android:src="@drawable/bg_general_day"
    android:visibility="visible" />

  <SurfaceView
    android:id="@+id/surface_view"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#00000000"
    android:visibility="visible" />

</RelativeLayout>
```

En outre, nous devons définir un thème particulier pour Flying Bird afin qu'il puisse s'afficher en plein écran. Notre thème va hériter de *Theme.AppCompat.Light.NoActionBar* permettant déjà de ne pas afficher de barre d'actions. A ce thème, nous ajoutons la propriété *android:windowFullscreen* positionnée à true afin de dire à l'OS que notre application devra être affichée en plein écran :



1 *Sprites de Flying Bird*

```
<style name="AppTheme.FullScreen" parent="Theme.AppCompat.Light.NoActionBar">
  <item name="android:windowFullscreen">true</item>
</style>
```

Le thème étant ensuite appliqué au niveau de l'application au sein de l'Android Manifest comme de coutume.

Initialisation du jeu

A la création de l'activité principale du jeu, on va récupérer les composants *ImageView* et *SurfaceView* définis précédemment dans le layout. On appelle ensuite la méthode *setKeepScreenOn* avec true en paramètre afin de spécifier que l'on souhaite garder l'écran allumé durant une partie de Flying Bird. On continue la configuration de notre instance de *SurfaceView* en définissant que l'activité principale du jeu, qui implémente l'interface OnTouchListener, recevra les évènements déclenchés lorsqu'un joueur touchera l'écran.

Après cela, on récupère l'objet *SurfaceHolder* associé à notre *SurfaceView* via un appel à la méthode *getHolder*. Cet objet nous permettra de pouvoir interagir avec la zone de dessin de la *SurfaceView* en accédant notamment à son élément *Canvas*. L'initialisation du jeu se poursuit en définissant le format des pixels pour le *SurfaceHolder* ainsi que l'implémentation de l'interface *SurfaceHolder.Callback* qui devra recevoir des appels lors des évènements liés au cycle de vie du premier objet nommé. Ces évènements étant la création, la destruction ou un changement affectant le *SurfaceHolder*.

Enfin, on termine en chargeant les différentes ressources utiles à Flying Bird telles que les tuyaux ou encore les pièces. On crée également une instance d'objet *SoundPool* qui servira par la suite pour gérer la partie son du jeu. On en profite également pour charger les différents sons que nous utiliserons durant une partie de Flying Bird depuis le répertoire assets. Pour cela, on ouvre le fichier de chaque son grâce à l'*AssetManager* avant de charger le contenu du fichier via la méthode *load* du *SoundPool*. Tout ceci nous donne le code d'initialisation suivant :

```
public class GameActivity extends AppCompatActivity implements Callback,
        OnTouchListener {
// …

@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_game);
  ivBackground = (ImageView) findViewById(R.id.iv_background);
  surfaceView = (SurfaceView) findViewById(R.id.surface_view);
  surfaceView.setKeepScreenOn(true);
  holder = surfaceView.getHolder();
  surfaceView.setZOrderOnTop(true);
  surfaceView.setOnTouchListener(this);
  holder.addCallback(this);
  holder.setFormat(PixelFormat.TRANSLUCENT);
  loadRes();
}

private void loadRes() {
  Resources res = getResources();
  blockerUp = res.getDrawable(R.drawable.img_block_up);
  blockerDown = res.getDrawable(R.drawable.img_block_down);
  coin = res.getDrawable(R.drawable.img_coin);
  soundPool = new SoundPool(5, AudioManager.STREAM_MUSIC, 0);
  AssetManager assetManager = res.getAssets();
  soundIds = new int[5];
```

```
 try {
  soundIds[SOUND_DIE] = soundPool.load(
     assetManager.openFd("sfx_die.ogg"), 1);
  soundIds[SOUND_HIT] = soundPool.load(
     assetManager.openFd("sfx_hit.ogg"), 1);
  soundIds[SOUND_POINT] = soundPool.load(
     assetManager.openFd("sfx_point.ogg"), 1);
  soundIds[SOUND_SWOOSHING] = soundPool.load(
     assetManager.openFd("sfx_swooshing.ogg"), 1);
  soundIds[SOUND_WING] = soundPool.load(
     assetManager.openFd("sfx_wing.ogg"), 1);
 } catch (IOException e) {
 }
}
```

Game Loop

L'évolution de tout jeu en informatique est réalisée au sein de ce qu'on appelle communément la Game Loop. Dans le cadre de l'implémentation de notre jeu Flying Bird, nous allons la mettre en place en créant un Thread séparé du Thread de rendu de l'IHM de l'application sous Android. Ce Thread séparé aura en charge la gestion de la Game Loop. Ainsi, une fois l'objet *SurfaceHolIl y a de cela 4 ans, la folie Flappy Bird s'emparait de l'univers des applications mobiles. Un simple jeu 2D réalisé par un développeur indépendant atteignait la première place des classements de l'App Store d'Apple et du Play Store de Google. Ce succès aura démontré qu'un jeu simple bien conçu au fort pouvoir addictif pouvait connaître un immense succès. Dans cet article, nous vous proposons de créer un clone de Flappy Bird sous Android afin de découvrir comment créer un jeu 2D simple.der* créé, on va démarrer notre Thread dédié que l'on nommera *DrawingThread*. Ce Thread sera arrêté lorsque le callback *surfaceDestroyed* sera appelé nous signifiant que l'instance d'objet *SurfaceHolder* est détruite. Ceci se produisant lorsque l'application est mise en pause par exemple ou arrêtée.

Au sein de notre *DrawingThread*, nous allons boucler tant que le Thread sera actif. Dans un premier temps, on va s'occuper du rendu de notre jeu. Pour ce faire on obtient une référence au *Canvas* de notre objet *SurfaceHolder* en appelant sa méthode *lockCanvas*. On va ensuite vider le contenu de ce *Canvas* avant d'itérer sur l'ensemble des éléments de notre jeu que nous souhaitons rendre à l'écran. Ces éléments étant nos fameux Sprites dont nous avons parlé précédemment. Cette itération est réalisée via un objet *Iterator* puisque l'on profite de ce parcours pour supprimer les Sprites considérés comme n'étant plus en vie. Ce travail est encapsulé au sein d'un bloc try / finally. Dans la partie finally, on va demander à ce que les mises à jour que nous avons effectuées sur le *Canvas* soit postées sur le *SurfaceHolder* via un appel à la méthode *unlockCanvasAndPost* avec l'instance courante de *Canvas* passée en paramètre.

On obtient alors un code ayant l'allure suivante :

```
// …
@Override
 public void surfaceCreated(SurfaceHolder holder) {
  surfaceCreated = true;
  startDrawingThread();
 }

 @Override
 public void surfaceDestroyed(SurfaceHolder holder) {
  surfaceCreated = false;
  stopDrawingThread();
 }

 private void startDrawingThread() {
  stopDrawingThread();
  drawingThread = new DrawingThread();
  drawingThread.start();
 }

 private void stopDrawingThread() {
  if (drawingThread != null) {
   drawingThread.interrupt();

   try {
     drawingThread.join();
   } catch (InterruptedException e) { }

   drawingThread = null;
  }
 }

 private class DrawingThread extends Thread {
  @Override
  public void run() {
   super.run();
   workSprites = new LinkedList<Sprite>(sprites);

   while (!Thread.interrupted()) {
    long startTime = System.currentTimeMillis();
    Canvas canvas = holder.lockCanvas();

    try {
     cleanCanvas(canvas);
     Iterator<Sprite> iterator = workSprites.iterator();

     while (iterator.hasNext()) {
      Sprite sprite = iterator.next();

      if (sprite.isAlive()) {
       sprite.onDraw(canvas, globalPaint, currentStatus);
      } else {
       iterator.remove();
      }
     }
    } finally {
     holder.unlockCanvasAndPost(canvas);
    }

    long duration = (System.currentTimeMillis() - startTime);
    long gap = GAP - duration;
```

```
    if (gap > 0) {
      try {
        sleep(gap);
      } catch (Exception e) {
        break;
      }
    }

    // … mises à jour de l'environnement du jeu

  } // fin du while
 }
}
```

Vous aurez également remarqué que l'on mesure le temps de rendu avant de comparer ce temps à une constante nommée GAP. Cette constante nous permet de rajouter, si nécessaire, une temporisation pour éviter que la phase de rendu de la Game Loop soit trop rapide.

Gestion des Sprites

La Game Loop mise en place avec la phase de rendu du jeu réalisée au sein d'un Thread dédié, nous allons pouvoir passer à la gestion des Sprites. Ces derniers ayant en commun un certain nombre de comportements, nous allons créer une interface pour modéliser ces comportements. L'interface *Sprite* proposera ainsi une méthode *onDraw* permettant de demander le dessin sur le *Canvas* du Sprite, une méthode *isAlive* pour savoir si un Sprite est toujours en vie ou non, une méthode *isHit* permettant de gérer la collision entre un Sprite et un autre Sprite et enfin une méthode *getScore* retournant le nombre de points associés à un Sprite :

```
public interface Sprite {

 void onDraw(Canvas canvas, Paint globalPaint, int status);
 boolean isAlive();
 boolean isHit(Sprite sprite);
 int getScore();

}
```

La méthode *isAlive* d'un Sprite est utile dans la Game Loop comme nous l'avons vu précédemment afin de savoir si on doit le rendre à l'écran ou bien si on doit le retirer de la liste des Sprites.
Nous commençons par créer une première implémentation de l'interface *Sprite*. La classe *BirdSprite* va ainsi être en charge de la gestion de l'oiseau dans notre jeu. Au sein du constructeur de l'objet *BirdSprite*, on charge les différentes versions visuelles de l'oiseau que nous utiliserons. L'oiseau pourra ainsi changer de couleur au fur et à mesure de la progression du joueur. On charge également les constantes permettant de calculer les dimensions de l'oiseau, sa position initiale à l'écran, son accélération ou encore la position du sol. Cette dernière constante nous permettant de détecter si l'oiseau a heurté le sol notamment.
Le gros du travail se situe au sein de la méthode *onDraw* du *BirdSprite* au sein de laquelle on va tout d'abord gérer le changement de couleur de l'oiseau via la propriété count. Si le statut du Sprite est différent de la constante *STATUS_NOT_STARTED*, on va incrémenter la position courante en hauteur de l'oiseau avec la vitesse courante chargée précédemment. Cette dernière étant ensuite incrémentée avec l'accélération sensée reproduire l'effet de la gravité sur notre oiseau. On s'assure ensuite que l'oiseau reste bien dans les limites de l'écran en remettant sa position courante à 0 si celle-ci venait à atteindre le haut de l'écran. On fait de même pour sa position en bas de l'écran. Enfin, on rend le Sprite associé à l'oiseau sur l'instance de *Canvas* passée en entrée de la méthode *onDraw*. Un dernier mot concernant l'implémentation de la méthode *isHit* pour laquelle on renverra true si la position courante de l'oiseau dépasse celle du sol représentée par la propriété *maxHeight*. Dans ce cas de figure, la partie sera terminée.
Cela nous donne le code suivant pour notre classe *BirdSprite* :

```
public class BirdSprite implements Sprite {

 private static final int[][] DRAWABLE_BIRD = new int[][] {
  new int[] { R.drawable.img_bird_blue_1, R.drawable.img_bird_blue_2,
    R.drawable.img_bird_blue_3 },
  new int[] { R.drawable.img_bird_yellow_1,
    R.drawable.img_bird_yellow_2, R.drawable.img_bird_yellow_3 },
  new int[] { R.drawable.img_bird_red_1, R.drawable.img_bird_red_2,
    R.drawable.img_bird_red_3 } };
 private static Random random = new Random();
 private static final int FLY_COUNT = 6;
 private int count = 0;
 private Drawable birds[] = new Drawable[4];
 private final int X;
 private int width, height, currentHeight;
 private int birdHeight, birdWidth;
 private float currentSpeed, acceleration, tapSpeed;
 private int maxHeight, hitPaddingTop,
         hitPaddingBottom, hitPaddingRight, hitPaddingLeft;

 public BirdSprite(Context context) {
  Resources res = context.getResources();
  int currentBird = random.nextInt(DRAWABLE_BIRD.length);
  birds[0] = birds[2] = res.getDrawable(DRAWABLE_BIRD[currentBird][0]);
  birds[1] = res.getDrawable(DRAWABLE_BIRD[currentBird][1]);
  birds[3] = res.getDrawable(DRAWABLE_BIRD[currentBird][2]);

  birdHeight = ViewUtil.dipResourceToPx(context, R.dimen.bird_height);
  birdWidth = birdHeight * birds[0].getIntrinsicWidth() / birds[0].getIntrinsicHeight();
  width = ViewUtil.getScreenWidth(context);
  height = ViewUtil.getScreenHeight(context);
  int xPosition = ViewUtil.dipResourceToPx(context, R.dimen.bird_position_x);
  X = width / 2 - birdWidth / 2 - xPosition;
  currentHeight = height / 2 - birdHeight / 2;

  // Chargement des constantes
  acceleration = ViewUtil.dipResourceToFloat(context,
    R.dimen.bird_acceleration);
  tapSpeed = ViewUtil.dipResourceToFloat(context, R.dimen.bird_tap_speed);
  maxHeight = height - ViewUtil.dipResourceToPx(context, R.dimen.ground_height);
  hitPaddingBottom = ViewUtil.dipResourceToPx(context,
    R.dimen.bird_hit_padding_bottom);
  hitPaddingTop = ViewUtil.dipResourceToPx(context,
```

```
    R.dimen.bird_hit_padding_top);
  hitPaddingLeft = ViewUtil.dipResourceToPx(context,
    R.dimen.bird_hit_padding_left);
  hitPaddingRight = ViewUtil.dipResourceToPx(context,
    R.dimen.bird_hit_padding_right);
  currentSpeed = 0;
 }

 public int getHitLeft() {
  return X + hitPaddingLeft;
 }

 public int getHitTop() {
  return currentHeight + hitPaddingTop;
 }

 public int getHitBottom() {
  return currentHeight + birdHeight - hitPaddingBottom;
 }

 public int getHitRight() {
  return X + birdWidth - hitPaddingRight;
 }

 @Override
 public void onDraw(Canvas canvas, Paint globalPaint, int status) {
  if (count >= 4 * FLY_COUNT) {
   count = 0;
  }

  if (status != Sprite.STATUS_NOT_STARTED) {
   currentHeight += currentSpeed;

   synchronized (this) {
    currentSpeed += acceleration;
   }
  }

  if (currentHeight <= 0) {
   currentHeight = 0;
  }

  if (currentHeight + birdHeight > maxHeight) {
   currentHeight = maxHeight - birdHeight;
  }

  Drawable bird = null;

  if (status == Sprite.STATUS_GAME_OVER) {
   bird = birds[0];
  } else {
   bird = birds[(count++) / FLY_COUNT];
  }

  bird.setBounds(X, currentHeight, X +
    birdWidth, currentHeight + birdHeight);
  bird.draw(canvas);
 }

 @Override
 public boolean isAlive() {
  return true;
 }

 @Override
 public boolean isHit(Sprite sprite) {
  return currentHeight + birdHeight >= maxHeight;
 }

 public void onTap() {
  synchronized (this) {
   currentSpeed = tapSpeed;
  }
 }

 @Override
 public int getScore() {
  return 0;
 }

 public int getX() {
  return X;
 }
}
```

Outre le *BirdSprite*, nous allons ajouter un *BlockerSprite* pour gérer les tuyaux s'affichant à l'écran qui doivent être évités par le joueur, un *CoinSprite* pour afficher les pièces devant être collectées ou encore un *GroundSprite* pour gérer le sol et son défilement horizontal pour donner l'illusion de la progression de l'oiseau à l'écran. Ces derniers sont bâtis sur le même principe avec un constructeur au sein duquel on initialise le Sprite avant d'implémenter les différentes méthodes de l'interface Sprite parmi lesquelles la méthode *onDraw* permettant de rendre le Sprite à l'écran.

Nous pouvons maintenant compléter notre Game Loop avec les mises à jour du jeu concernant les Sprites. On rajoutera ainsi le code suivant pour détecter si le jeu doit s'arrêter car le joueur a perdu :

```
boolean hit = false;

for (Sprite sprite : workSprites) {
 if (sprite.isHit(birdSprite)) {
  onHit();
  hit = true;
  break;
 }
}

if (hit) {
 workSprites.addLast(splashSprite = new SplashSprite());
 currentStatus = Sprite.STATUS_GAME_OVER;
 continue;
}
```

Ici, on parcourt les Sprites et si le *BirdSprite* a heurté le sol ou un tuyau, on passe le statut courant du jeu à *STATUS_GAME_OVER* : la partie est donc terminée et on devra afficher au joueur l'écran de fin au prochain passage de la Game Loop. Cet écran étant modélisé au sein d'un objet *GameOverSprite*.

Interaction avec le joueur

Le but de Flying Bird étant de permettre au joueur de faire progresser l'oiseau en tapotant sur l'écran, nous devons agir au sein de la méthode *onTouch* de l'interface *OnTouchListener* dont notre activité principale hérite. C'est dans cette méthode que nous allons interagir avec le joueur lorsqu'il tapera à l'écran. Si le jeu n'est pas encore démarré, une première tape sur l'écran va permettre de passer son statut à la constante STATUS_NORMAL ce qui autorisera la mise à jour des éléments du jeu au sein de la Game Loop où la condition suivante est définie au début de la zone de mise à jour :

```
if (currentStatus == Sprite.STATUS_NOT_STARTED) {
  continue;
}
```

Dans le cas où la partie est en cours, on va appeler la méthode *onTap* du *BirdSprite* afin d'incrémenter sa vitesse courante ce qui aura pour effet de faire monter l'oiseau à l'écran. Cette montée à l'écran de l'oiseau venant lutter contre l'effet de la gravité en évitant qu'il ne chute vers le sol.

Enfin, si la partie est terminée, on va appeler la méthode *onTap* de l'objet *GameOverSprite* qui nous renverra une constante en sortie permettant de savoir quelle partie de l'écran de fin le joueur à toucher afin de réagir en conséquence soit pour démarrer une nouvelle partie soit pour partager le score réussi via les réseaux sociaux par exemple. L'implémentation de la méthode *onTouch* a donc l'allure suivante :

```
@Override
public boolean onTouch(View v, MotionEvent event) {
 if (event.getAction() == MotionEvent.ACTION_DOWN) {
  switch (currentStatus) {
   case Sprite.STATUS_NOT_STARTED:
    currentStatus = Sprite.STATUS_NORMAL;
   case Sprite.STATUS_NORMAL:
    birdSprite.onTap();
    soundPool.play(soundIds[SOUND_WING], 0.5f, 0.5f, 1, 0, 1);
    break;
   case Sprite.STATUS_GAME_OVER:
    switch (gameOverSprite.onTap((int) event.getX(), (int) event.getY())) {
     case PLAY:
      playSwooshing();
      restart();
      break;
     case SHARE:
      share();
      break;
     default:
      break;
    }
    break;
   default:
    break;
  }
 }

 return false;
}
```

Gestion des obstacles

Dans la partie concernant la gestion des Sprites, la méthode *isHit* a déjà été évoquée laissant clairement deviner qu'elle servirait dans la gestion des obstacles que notre cher oiseau doit éviter tout au long du jeu. Pour certains Sprites, cette méthode n'a pas de sens et ils renverront ainsi toujours false à l'appel de cette méthode. C'est le cas par exemple d'un *CoinSprite*. En revanche, cette méthode prend tout son sens pour les objets de type *BlockerSprite* représentant les tuyaux que l'oiseau doit éviter.

Nous allons donc nous intéresser plus en détails à l'implémentation de la méthode *isHit* au sein de la classe *BlockerSprite*. La méthode *isHit* prend en entrée un objet de type *Sprite* qui correspond à l'objet pour lequel nous voulons déterminer s'il est entré en collision avec le *BlockerSprite* courant. Dans un premier temps nous vérifions donc que le Sprite passé en entrée est bien de type *BirdSprite* puisque nous ne souhaitons gérer que les collisions entre l'oiseau et les tuyaux. Si c'est bien le cas, on détermine le rectangle englobant l'oiseau avant de le comparer à la position du tuyau courant. S'il y a collision, on renvoie alors true. Au sein de la Game Loop, ce retour permettra de savoir si le jeu peut continuer ou s'il est terminé. Le code de la méthode *isHit* du *BlockerSprite* se présente comme suit :

```
@Override
public boolean isHit(Sprite sprite) {
 if (sprite instanceof BirdSprite) {
  BirdSprite b = (BirdSprite) sprite;
  int bTop = b.getHitTop();
  int bBottom = b.getHitBottom();
  int bRight = b.getHitRight();
  int bLeft = b.getHitLeft();
  int left = (int) currentX;
  int right = (int) currentX + blockWidth;
  return (bTop < currentUpHeight || bBottom > currentUpHeight + gap)
      && ((bRight > left && bRight < right) || (bLeft > left && bLeft < right));
 } else {
  return false;
 }
}
```

Rajout des sons

Afin de rendre notre clone de Flappy Bird plus attractif, il est essentiel de reproduire les sons présents dans le jeu original. Ces sons se produisent notamment au démarrage du jeu, lorsque l'oiseau marque un point ou lorsque l'utilisateur tape l'écran et que l'oiseau prend alors de la hauteur. Nous avons pu voir dans la partie d'initialisation du jeu que les sons étaient chargés au sein d'un tableau d'entiers nommé *soundIds* et indexé avec des constantes allant de 0 à 4. Chacune représentant un son bien particulier à jouer au cours de la partie de Flying Bird. Les sons sont joués en appelant

la méthode *play* de l'objet *SoundPool* créé au début du jeu. L'émission d'un son lorsque l'oiseau va percuter un tuyau se fera de la sorte au sein d'une méthode *onHit* dédiée :

```
private void onHit() {
 runOnUiThread(new Runnable() {
  @Override
  public void run() {
   if (!isFinishing()) {
    soundPool.play(soundIds[SOUND_HIT], 0.5f, 0.5f, 1, 0, 1);
   }
  }
 });
}
```

Il est bon de noter ici que le son est émis dans le Thread principal de l'application. L'appel à la méthode *play* se doit donc d'être imbriqué au sein de la méthode *runOnUiThread*.

Gestion des scores

Le succès phénoménal de Flappy Bird est dû également au côté addictif créé par la compétition entre les joueurs de par le monde. Cette compétition n'ayant pu être engendrée que par la comparaison des scores entre les joueurs. Gérer les scores dans ce type de jeu s'avère donc fondamental. Dans le cadre de Flying Bird, nous nous limiterons à un stockage des scores en local en mettant à profit l'API Preferences proposée par Android en standard.

En ce sens, nous créons un objet *Scores* proposant des méthodes statiques pour obtenir le meilleur score réalisé et enregistrer un nouveau meilleur score. Voici son code :

```
public class Scores {

 private static final String PREF_DEFAULT = "com.ssaurel.flyingbird.PREF_DEFAULT";
 private static final String HIGH_SCORE = "high_score";
 public static boolean NEW_BEST_SCORE = false;

 public static int highScore(Context context) {
  SharedPreferences p = context.getSharedPreferences(PREF_DEFAULT,
          Context.MODE_PRIVATE);
  return p.getInt(HIGH_SCORE, 0);
 }

 public static void storeHighScore(Context context, int score) {
  NEW_BEST_SCORE = true;
  SharedPreferences p = context.getSharedPreferences(PREF_DEFAULT,
          Context.MODE_PRIVATE);
  p.edit().
   putInt(HIGH_SCORE, score).
   commit();
 }
}
```

Flying Bird en action

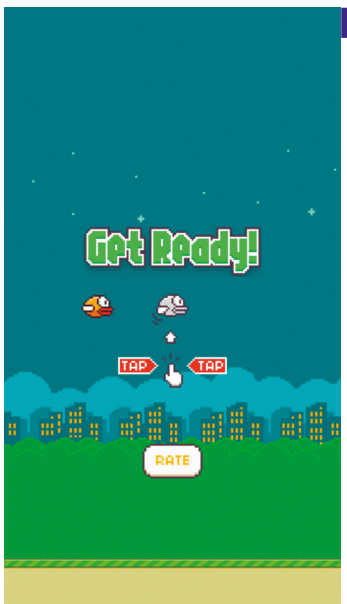
Le code de notre jeu terminé, il est temps de passer à la partie la plus agréable, à savoir tester notre clone de Flying Bird en conditions réelles sur smartphone Android. La figure 2 présente l'écran initial du jeu.

Après un premier toucher sur l'écran, le jeu démarre et il faut essayer de faire progresser l'oiseau en évitant les tuyaux, tout en collectant le maximum de pièces pour marquer le plus de points possibles (figure 3).

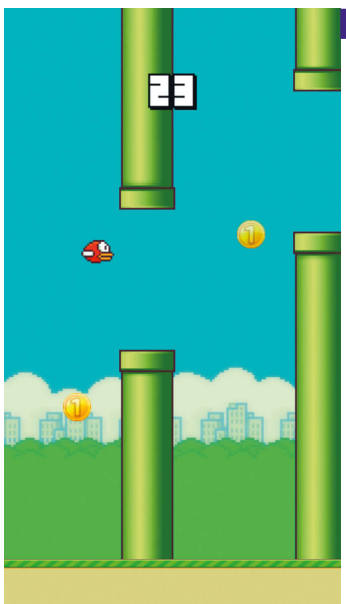
Enfin, si l'oiseau percute un tuyau ou touche le sol, la partie est terminée et l'écran de fin de jeu est affiché (figure 4).

Conclusion

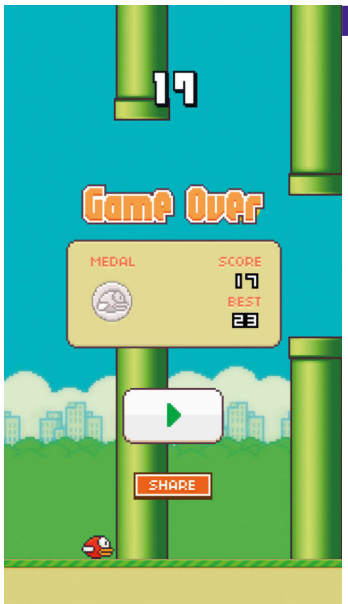
Au travers de la réalisation d'un clone du célèbre jeu Flappy Bird, cet article vous aura permis de découvrir avec quelle simplicité il est possible de créer un petit jeu en 2D de type casual sous Android. Bien entendu, un nombre important de bibliothèques existe pour faciliter la création de ce type de jeux mais il est toujours intéressant de découvrir comment ils sont réalisés avant de recourir à des frameworks. Maintenant que vous avez acquis les bases essentielles dans la création de jeux 2D sous Android, vous pouvez passer à la réalisation de votre propre jeu dans l'espoir de créer, pourquoi pas, le prochain Flappy Bird. C'est tout le mal que l'on vous souhaite.•



2 *Ecran de démarrage de Flying Bird*



3 *Flying Bird en action*

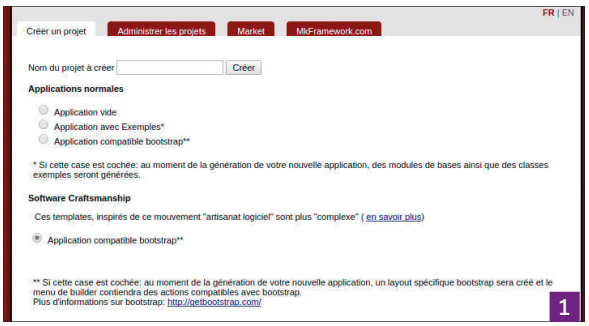


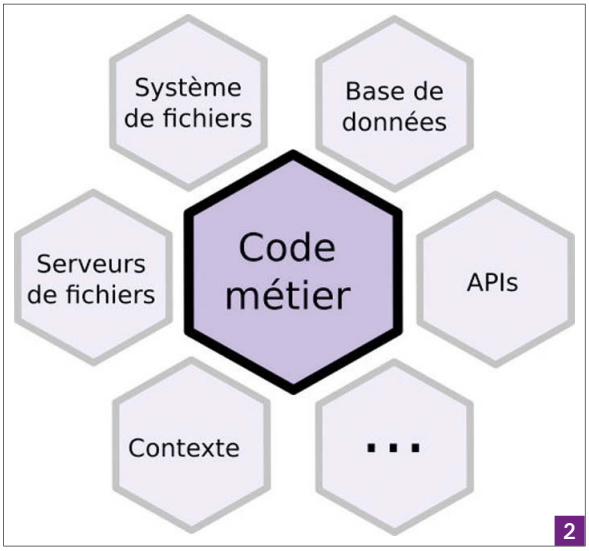
4 *Ecran de fin de Flying Bird*

Michaël Bertocchi
Créateur du mkframework
@dupot_org

Créer une application software craftsmanship avec mkframework

Vous avez peut-être entendu parler ces derniers temps d'un mouvement qui vise à redorer le blason de notre profession en l'assimilant à celui de l'artisanat, le : "software craftsmanship". Dans le même esprit qu'un artiste, nous souhaitons de plus en plus que l'utilisateur final soit aussi content du resultat visible, que nos pairs/collègues de la conception/implémentation.
Et contrairement à une création artisanale, nos applications sont amenées à être maintenues, retravaillées, améliorées, sécurisées, donc autant s'evertuer à faciliter ce travail post livraison, non ?
Pour suivre ce mouvement, mkframework propose depuis mars 2017 un nouveau groupe de template dans cette philosophie.





Le nouveau template dans le builder du mkframework

Je vais ici présenter l'implémentation d'un nouveau template dans cette philosophie sur le mkframework (http://mkframework.com)
Quand vous ouvrez le builder, vous avez un nouveau template de génération proposé [figure 1]

Présentation des spécificités de cette gamme de template

Pour ceux qui ne l'ont jamais utilisé, mkframework, lors de la création d'une application vous accompagne dans le démarrage de celle-ci (création de la couche modèle, de modules d'authentification, gestion de droits, CRUD…)
Cette nouvelle gamme de template n'échappe pas à cette règle, mais on distingue cependant les points suivants :

- Utilisation d'une architecture hexagonale ;
- Multilingue intégré dès le départ ;
- Génération des tests unitaires associés ;
- Modules avec héritage.

Quelques mots sur l'architecture hexagonale

L'idée est de découpler totalement le code de tout élément extérieur [figure 2], pour cela au lieu, comme sur les autres templates, d'écrire une partie du code dans le module (équivalent du contrôleur sur les autres frameworks) ou model, on réunit ici le code dit "métier" dans une classe "business".
Par exemple dans un module CRUD* pour une table auteur, on passe d'un code qui gérait tout comme par exemple :

```
private function processSave(){
//si ce n'est pas une requête POST on ne soumet pas
  if(!_root::getRequest()->isPost() ){
      return null;
  }

$oPluginXsrf = new plugin_xsrf();
//on verifie que le token est valide
if(!$oPluginXsrf->checkToken( _root::getParam('token') ) ){
return array('token' => $oPluginXsrf->getMessage() );
}

$iId = module_auteur::getParam('id',null);
if($iId == null){
$oAuteur = new row_auteur();
  }else{
      $oAuteur = model_auteur::getInstance()->findById( _root::getParam('id',null) );
  }

  $tColumn = array('nom','prenom','email');
  foreach($tColumn as $sColumn){
```

```
    $oAuteur->$sColumn = _root::getParam($sColumn,null) ;
}


if($oAuteur->save() ){
    //une fois enregistré on redirige (vers la page liste)
    _root::redirect('auteur::list');
}else{
    return $oAuteur->getListError();
}

}
```

Note

La vérification de la cohérence de données était faite dans la classe modèle model_auteur (méthode getCheck pour ceux qui connaissent).

A un code qui fait appel à une classe business :

```
private function processSave() {
//si ce n'est pas une requete POST on ne soumet pas
if (!_root::getRequest()->isPost() ) {
return null;
}

$oPluginXsrf = new plugin_xsrf();
//on verifie que le token est valide
if (!$oPluginXsrf->checkToken(_root::getParam('token')) ) {
return array('token' => $oPluginXsrf->getMessage());
}

$tParams = _root::getRequest()->getParams();

$oBusiness = new business_crudAuteur(model_auteur::getInstance(), _root::getI18n
(), new plugin_sc_valid() );

$id = _root::getParam('id', null);
if ($id == null) {
if (false === $oBusiness->insertItem(new row_auteur, $tParams)) {
return $oBusiness->getReturn()->getData('tError');
}
} else {
if (false === $oBusiness->updateItem(_root::getParam('id'), $tParams)) {
return $oBusiness->getReturn()->getData('tError');
}
}

//une fois enregistré on redirige (vers la page liste)
_root::redirect('auteur::list');
}
```

Note

On n'a volontairement pas mis la partie token (sécurité xsrf) car ceci n'est pas du code "métier". C'est lié au formulaire web, le code métier doit être totalement neutre de son environnement d'execution, ainsi on pourra aussi bien appeler cette classe business pour un site web, un webservice, un batch ou autre.

Ici on fait la partie vérification de la cohérence et enregistrement dans la classe business.

Voici ici le constructeur de la classe business (qui serait généré par le builder qui ressemblerait à ceci) :

```
<?php

class business_crudAuteur extends business_abstract {

  protected $_oModel;
  protected $_oAuth;
  protected $_oI18n;
  protected $_oValid;
  protected $_tColumn = array('nom','prenom','email',);

  public function __construct(interface_model $oModel_, interface_i18n $oI18n_,
interface_valid $oValid_) {
      $this->_oModel = $oModel_;
      $this->_oI18n = $oI18n_;
      $this->_oValid = $oValid_;
  }
  (...)
```

Dans la logique d'architecture hexagonale, on donne à cette classe les éléments de contexte respectant le contrat d'interface: classe modèle pour agir avec les données, classe de langue pour gérer les traductions et classe de validation de données (pour valider celle-ci avant de les insérer en base de données).
L'idée d'injecter ces éléments de contexte permet trois choses :

- De pouvoir changer de classe si besoin ;
- De pouvoir déléguer l'écriture de certaines classes (grâce au contrat d'interface) ;
- De pouvoir facilement mettre en place des tests unitaires.

Un autre extrait intéressant est celui de la méthode de mise à jour de l'enregistrement:

```
public function updateItem($id_, $tParam_ ) {

    $oValid = $this->getCheck($tParam_ );
  if (!$oValid->isValid()) {
    return $this->sendReturn(false, array('tError' => $oValid->getListError()));
    }

    $oAuteur = $this->_oModel->findById($id_ );
  foreach ($this->_tColumn as $sColummn) {
        $oAuteur>$sColummn = $tParam_[$sColummn];
    }

    $this->_oModel->update($oAuteur);

  return true;
  }
```

Et voici la méthode appelée pour vérifier les données avant de les enregistrer :

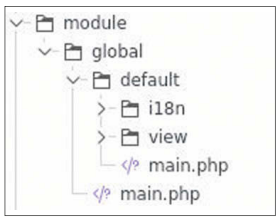
```
public function tr($sTag_){
  return $this->_oI18n->tr($sTag_);
  }

  //vous pouvez ajouter ici des verifications supplémentaires
public function getCheck($tParam_ ) {
    $this->_oValid->load($tParam_ );

    foreach ($this->_tColumn as $sColumn ) {
```

```
        $this->_oValid->isNotEmpty($sColumn, $this->tr('errorIsEmpty') );
    }
    $this->_oValid->isEmailValid('email', $this->tr('errorIsEmailInvalid') );

return $this->_oValid;
}
```

*CRUD Create Read Update Delete: module permettant d'ajouter,modifier,afficher,supprimer et lister des enregistrements en base.

Des applications multi-lingue par défaut

Il est difficile de modifier une application pour la rendre multi-lingue ; ici dès la création de l'application par le builder, elle l'est.
Vous avez des fichiers de langue à la racine de votre projet : data/i18n/fr.php



Ainsi que dans chacun des modules, permettant de gérer vos traductions par héritage: [figure 3]
Pour l'utiliser dans vos vues ou ailleurs, c'est très simple, il suffit d'utiliser la fonction tr()

```
<?php echo tr('annuler') ?>
```

Les tests unitaires

Toujours dans l'esprit d'artisan du code, un des points est de pouvoir améliorer son code. Ceci passe souvent par de la réécriture, du refactoring, et pour cela il est nécessaire de s'assurer de la continuité de fonctionnement de l'application, d'où l'importance de générer et d'écrire des tests unitaires.
Le builder, génère donc désormais pour ce nouveau template, en plus du code fonctionnel nécessaire (module, model, views), les tests unitaires associés, qu'il vous faudra bien entendu enrichir en fonction de vos besoins.
Voici par exemple un extrait de test unitaire généré pour notre module CRUD de notre table article :

```
public function test_insertItemShouldReturnErrorsMissing() {
        $oMockModel = $this->getMock('interface_model');

//instanciation de la classe business avec les éléments de contexte,
//notamment un mock pour la classe modèle
    $oBusiness = new business_crudAuteur($oMockModel,new plugin_i18nFake(), new
plugin_sc_valid() );

        $tParam = array();
        $tColumn = array('nom','prenom','email');
    foreach ($tColumn as $sColumn) {
            $tParam[$sColumn] = null;
        }

        $bReturn = $oBusiness->insertItem(new stdclass(), $tParam);

        $this->assertEquals(false, $bReturn);

        $tError = array();
    foreach ($tColumn as $sColumn) {
            $tError[$sColumn] = array('{errorIsEmpty}');
        }

        $this->assertEquals($tError, $oBusiness->getReturn()->getData('tError'));
    }
```

Ici le test unitaire fait juste de vérifier qu'en passant toutes les propriétés à vide sur notre auteur, il a bien un retour d'erreur lors de la validation des données avec des messages de champs vide (errorIsEmpty).
Libre à vous d'enrichir ces tests unitaires pour ajouter des cas métier plus pertinents, vous avez déjà la classe et un exemple pour chaque méthodes générées.

Last but not least: l'héritage multi niveaux

Une dernière chose qui diffère sur ce template est la capacité d'héritage pour les modules : vous pouvez faire des modules héritant d'autres.
Par exemple, si vous avez un espace privé, vous pouvez faire un module "private" dont hériteront les modules qui le sont, ainsi :

- Vous y activerez l'authentification pour tous ses fils ;
- Vous pourrez y créer le layout commun ;
- Vous pourrez y charger le/les menus de navigation.

Ceci permet de faciliter le développement de vos modules qui partagent des mêmes attributs (partie privée, partie publique, partie admin, partie membre…).
Comme écrit précédemment vous avez également un héritage du fichier de langue : ainsi vous pouvez avoir des traductions génériques à la racine, puis ajouter des fichiers de langue propre à chacun des modules.

Conclusion

Voilà pour la présentation de ce nouveau template. Comme d'habitude, vous apprécierez le travail du builder qui vous permettra de gagner énormément de temps et vous permettra d'avancer et d'entrer dans le vif du sujet beaucoup plus rapidement.
Et vous pourrez apprécier cet autre mode de développement qui regroupe bien des avantages à terme.
Car même s'il demande un peu plus de temps à appréhender que les templates "normaux", ils ont l'avantage d'être pensés à plus long terme : c'est à dire à la maintenance et aux différentes évolutions qui pourront y être développées avec la sécurité procurée par les tests unitaires générés par le builder et enrichis par vos soins.
Une rubrique spécifique à ce nouveau template est disponible sur le site du mkframework : http://mkframework.com
Si vous appréciez ce mouvement, je vous invite à vous renseigner autour de chez vous sur d'éventuels meetups, présentations et également, pour les anglophiles, à lire la bible des amoureux du code bien écrit : Clean Code de Robert C. Martin. •

Adrien Clerbois
Developer @ Ingenico ePayments
Microsoft MVP Visual Studio and Development Technologies
@aclerbois

L'OpenAPI pour documenter votre ASP.NET Core 2.0 Web API

La spécification OpenAPI (anciennement Swagger Specification) est un format de description API pour les API REST. Un fichier OpenAPI vous permet de décrire l'ensemble de votre API, y compris :

- paramètres disponibles (/utilisateurs) et opérations sur chaque paramètre (GET/utilisateurs, POST/utilisateurs) ;
- paramètres de fonctionnement entrées et sorties pour chaque opération ;
- méthodes d'authentification ;
- coordonnées, licence, conditions d'utilisation et autres informations.

Les spécifications API peuvent être écrites en YAML ou JSON. Le format est facile à apprendre et lisible pour les humains et les machines. Plus d'informations sur :
https://github.com/OAI/OpenAPI-Specification/blob/master/versions/3.0.0.md
Aujourd'hui, il existe plusieurs moyens d'utiliser l'OpenAPI pour une application ASP.NET Core 2.0 Web Api. L'un de ceux-ci est Swagger (https://swagger.io/), dans cet article, nous allons explorer l'implémentation de Swagger dans une application ASP.NET Core 2.0.

Qu'est-ce que Swagger ?

Swagger est une représentation graphique de votre API RESTful. Swagger UI permet à quiconque - qu'il s'agisse de votre équipe de développement ou de vos clients finaux - de visualiser et d'interagir avec les ressources de l'API sans aucune logique d'implémentation en place. Il est automatiquement généré à partir de votre spécification Swagger, avec la documentation visuelle, ce qui facilite l'implémentation et la consommation côté client.

Ajouter Swagger à une application ASP.NET Core 2.0 Web API

Créer un projet ASP.NET Core 2.0 WebApi et installer le paquet de **NuGet Swashbuckle.AspNetCore**.
Le projet Swashbuckle est composé de trois paquets :

- **Swashbuckle.AspNetCore.Swagger :** middleware pour exposer les terminaux Swagger JSON des API basées sur ASP. NET Core.
- **Swashbuckle.AspNetCore.SwaggerGen :** Swagger Codegen génère la construction des Endpoints et des SDK clients à partir de vos spécifications OpenAPI.
- **Swashbuckle.AspNetCore.SwaggerUi :** middleware pour exposer une version embarquée de swagger-ui à partir d'une application ASP. NET Core.

Vous pouvez installer ces composants à partir du meta-paquet Swashbuckle.AspNetCore, soit à partir du [NuGet Package Manager], soit à partir du [Package Manager Console] (Commande : Install-Package Swashbuckle.AspNetCore) ou encore à partir de votre invite de commande (commande : dotnet add package Swashbuckle.AspNetCore).

Référencer Swagger dans notre projet

Par défaut dans un projet ASP.NET Core 2.0 WebApi, le contrôleur ValuesController est généré avec des méthodes d'action API pour des verbes http (GET, POST, PUT, DELETE).
Ouvrez le fichier Startup.cs pour ajouter à votre application le middleware Swagger.

```
public void ConfigureServices(IServiceCollection services)
{
  services.AddMvc();
  services.AddSwaggerGen(c =>
  {
    c.SwaggerDoc("v1", new Info
    {
      Version = "v1",
      Title = "Ma première API",
      Description = "API pour le magazine Programmez!",
      TermsOfService = "None",
      Contact = new Contact { Name = "Adrien Clerbois", Email = "adrien@aclerbois.
be", Url = "www.aclerbois.be" }
    });
  });
}
```

Il nous faut activer Swagger et SwaggerUi dans la méthode Configure()

```
public void Configure(IApplicationBuilder app, IHostingEnvironment env)
{
  if (env.IsDevelopment())
  {
    app.UseDeveloperExceptionPage();
  }

  app.UseMvc();
  app.UseSwagger();
  app.UseSwaggerUI(c =>
  {
    c.SwaggerEndpoint("/swagger/v1/swagger.json", "Ma première API");
  });
}
```

La dernière étape à réaliser est de changer l'URL de lancement de l'application pour que Swagger UI se charge au lancement. Pour définir Swagger UI comme URL de lancement, cliquez avec le bou-

Remarque

Tout cela est nécessaire pour configurer Swagger. Vous pouvez définir l'url où sera accessible le fichier swagger.json en modifiant la ligne 12.

ton droit de la souris sur [Projet] -> sélectionnez [propriétés] -> naviguez jusqu' à l'onglet [Debug]. Sur l'onglet [Debug], changez la valeur de Launch Browser en "swagger". 1

Dans le cas de l'utilisation de tout autre IDE, configurez-le pour que votre application web api se lance sur http://[applicationUrl]/swagger. Voici le résultat de la première exécution : 2

Swagger utilise des couleurs différentes pour chaque verbe HTTP afin de différencier les actions API. En cliquant sur n'importe quelle méthode, vous obtiendrez des détails sur les paramètres acceptés, le type de retour et vous pourrez tester la méthode.

Ceci est réalisé avec une configuration minimale et sans aucune personnalisation. Swagger peut être personnalisé pour inclure les commentaires XML de la méthode, en affichant les valeurs enum comme valeurs de chaîne et en générant différents documents Swagger pour différentes versions de l'API.

Aller plus loin dans la configuration

Par défaut, Swagger n'affiche pas les commentaires XML qui décorent les actions de contrôleur. Il y a une option qui permet de les afficher avec SwaggerUi.

D'abord, nous devons nous assurer que lors de la compilation du projet, tous les commentaires XML sont sauvés dans un fichier XML. Swagger utilisera ce fichier pour afficher les commentaires XML dans SwaggerUI. Dans Visual Studio, afin de sauvegarder les commentaires XML dans un fichier, cliquez droit sur le projet -> [properties] et naviguez sur l'onglet [build]. Dans cet onglet, activez l'option **"XML documentation file"**. 3

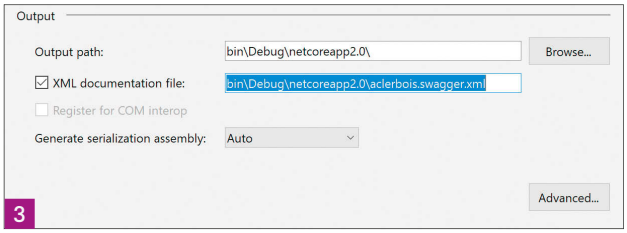
Vous pouvez obtenir le même résultat en modifiant le fichier du projet csproj :

```
<PropertyGroup Condition="'$(Configuration)|$(Platform)'=='Debug|AnyCPU'">
  <DocumentationFile>bin\Debug\netcoreapp2.0\aclerbois.swagger.xml</DocumentationFile>
</PropertyGroup>
```

En activant cette option, les commentaires xml sont enregistrés dans un fichier XML à la compilation de votre projet. Ce fichier est situé dans le dossier bin/[Debug/Release]\netcoreapp2.0. Nous devons passer ce chemin méthode IncludeXMLComments de Swashbuckle.

Ajoutez une méthode dans Startup.cs pour obtenir le chemin d'accès du XML généré. Ce code pour obtenir le chemin XML fonctionnera aussi bien dans votre environnement local que dans l'environnement de production.

```
private string GetXmlCommentsPath()
{
  var app = PlatformServices.Default.Application;
  return System.IO.Path.Combine(app.ApplicationBasePath, "aclerbois.swagger.xml");
}
```



3

Ensuite, utilisez cette méthode pour configurer SwaggerGen à prendre en compte le fichier xml :

```
public void ConfigureServices(IServiceCollection services)
{
  // do stuff
  services.AddSwaggerGen(c =>
  {
    c.SwaggerDoc(/*configuration de votre document swagger */);
    c.IncludeXmlComments(GetXmlCommentsPath());
  });
  // do stuff
}
```

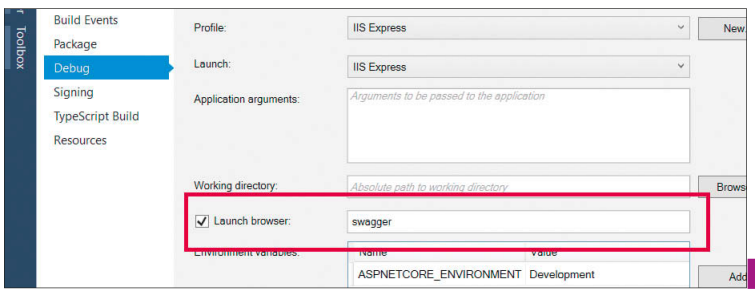
Décorons une des méthodes présentes dans le ValuesController :

```
/// <summary>
/// Retourne une valeur à partir de l'Id fourni
/// </summary>
/// <remarks>Voici une description pour la méthode</remarks>
/// <param name="id">Id d'une valeur</param>
/// <returns>Retourne une valeur</returns>
[HttpGet("{id}")]
public string Get(int id)
{
  return "value";
}
```

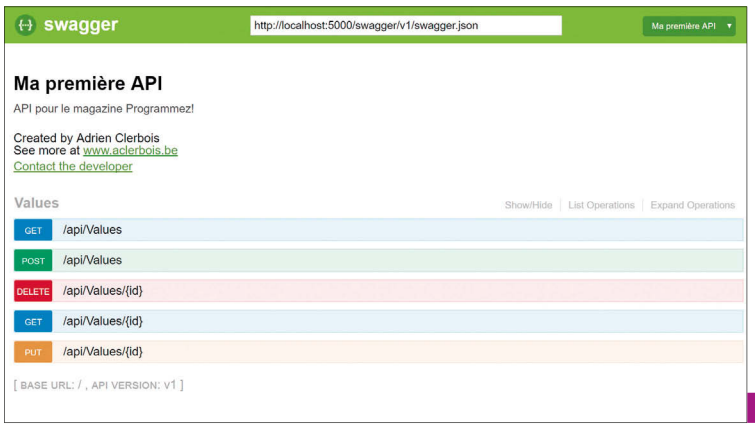
Vérifions le résultat : 4

Décrire les valeurs énumératives comme chaîne de caractères

SwaggerUI permet d'afficher des énumérations comme chaîne de caractères.



1



2

```
public void ConfigureServices(IServiceCollection services)
{
  // do stuff
  services.AddSwaggerGen(c =>
  {
    c.SwaggerDoc(/*configuration de votre document swagger */);
    c.IncludeXmlComments(GetXmlCommentsPath());
    c.DescribeAllEnumsAsStrings();
  });
  // do stuff
}
```

Pour tester ceci, ajoutez une nouvelle énumération dans votre projet :

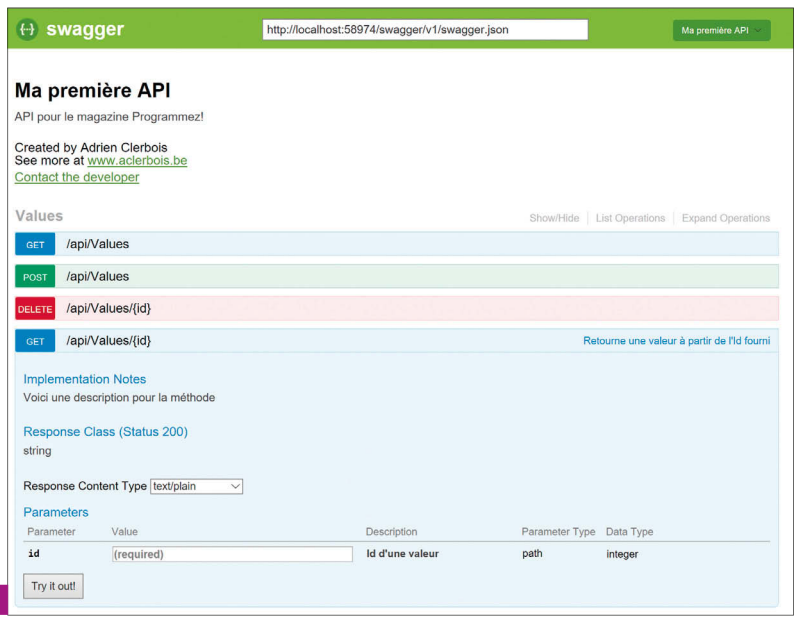
```
public enum Status{
  Active = 0,
  Disabled = 1
}
```

Implémentez une action avec l'énumération :

```
/// <summary>
/// Retourne une valeur à partir de l'Id fourni
/// </summary>
/// <remarks>Voici une description pour la méthode</remarks>
/// <param name="id">Id d'une valeur</param>
/// <param name="status">Status de la valeur</param>
/// <returns>Retourne une valeur</returns>
[HttpGet("{id}")]
public string Get(int id, Status status)
{
  return "value";
}
```

Et le résultat dans SwaggerUI : 5
Vous pouvez également utiliser la méthode DescribeStringEnumsInCamelCase qui permet d'obtenir les valeurs de l'énumération dans la case format CamelCase.

5



4

V1, V2, V3, … Restons rétro-compatible.

Le versioning permet de déployer les fonctionnalités en temps voulu, sans casser le système existant. Il peut également aider à fournir des fonctionnalités supplémentaires aux clients sélectionnés. La version de l'API peut être faite de différentes manières comme ajouter la version dans l'URL ou comme paramètre de chaîne de requête, via l'en-tête personnalisée et via Accept-Header.
Pour pouvoir utiliser le versionning dans l'ASP.NET Core Web Api, référencez le projet Microsoft.AspNetCore.Mvc.Versioning (Install-Package Microsoft.AspNetCore.Mvc.Versioning).
Une fois le paquet restauré, nous avons besoin de le configurer dans le fichier Startup.cs :

```
public void ConfigureServices(IServiceCollection services)
{
  // Add MVC and Swagger stuff
  services.AddApiVersioning(option =>
  {
    option.ReportApiVersions = true;
    option.AssumeDefaultVersionWhenUnspecified = true;
    option.DefaultApiVersion = new ApiVersion(1, 0);
  });
}
```

Dans cet exemple de code, le bloc de configuration AddApiVersionning implémente trois paramètres :

- **ReportApiVersions :** ceci est facultatif. Mais lorsqu'elle est définie sur true, l'API renvoie les informations des versions prises en charge dans l'en-tête de réponse.
- AssumeDefaultVersionWhenUnspecified : cette option sera utilisée pour servir la requête sans version. La version de l'API supposée par défaut serait 1.0.
- **DefaultApiVersion :** Cette option est utilisée pour spécifier la version par défaut de l'API à utiliser lorsqu'aucune version n'est spécifiée dans la requête. La version par défaut sera 1.0.

Afin de pouvoir définir la version d'un contrôleur, décorez celui-ci avec la décoration ApiVersion :

```
namespace aclerbois.swagger.Controllers.V1
{
  [ApiVersion("1.0")]
  [Route("api/[controller]")]
```

```
public class ValuesController : Controller
{
  [HttpGet]
  public IActionResult Get() => Ok(new string[] { "v1value1" });
}

namespace aclerbois.swagger.Controllers.V2
{
  [ApiVersion("2.0")]
  [Route("api/[controller]")]
  public class ValuesController : Controller
  {
    [HttpGet]
    public IActionResult Get() => Ok(new string[] { "v2value2" });
  }
}
```

Afin de cibler la version dans vos requêtes, il existe deux moyens :

- L'url : ajoutez dans le chemin de votre requête HTTP le paramètre ?api-version=[numero-de-version].
- Les entêtes http : dans l'en-tête http, utilisez api-version avec pour valeur la version souhaitée.

Référencer les versions dans Swagger

Après avoir ajouté le versionning de vos contrôleurs dans votre application ASP.NET MVC Core, vous devez créer la version v2 dans votre AddSwaggerGen.

```
public void ConfigureServices(IServiceCollection services)
{
  services.AddSwaggerGen(c =>
  {
    c.SwaggerDoc("v1", new Info
    {
      Version = "v1",
      Title = "Ma première API",
      Description = "API pour le magazine Programmez!",
      Contact = new Contact { Name = "Adrien Clerbois", Email = "adrien@aclerbois.
be", Url = "www.aclerbois.be" }
    });

    c.SwaggerDoc("v2", new Info
    {
      Version = "v2",
      Title = "Ma seconde version de mon API",
      Description = "API Programmez! V2",
      Contact = new Contact { Name = "Adrien Clerbois", Email = "adrien@aclerbois.
be", Url = "www.aclerbois.be" }
    });
  });
}
```

Nous devons également mettre à jour l'interface utilisateur pour afficher plusieurs paramètres. Ils seront listés dans le coin supérieur droit de la page, permettant aux utilisateurs de basculer entre les différents documents. Sur vos classes, et ce afin de faire le filtre par version, vous devez apposer le décorateur ApiExplorerSettings, ce décorateur peut être apposé à la fois sur le contrôleur ou une action :

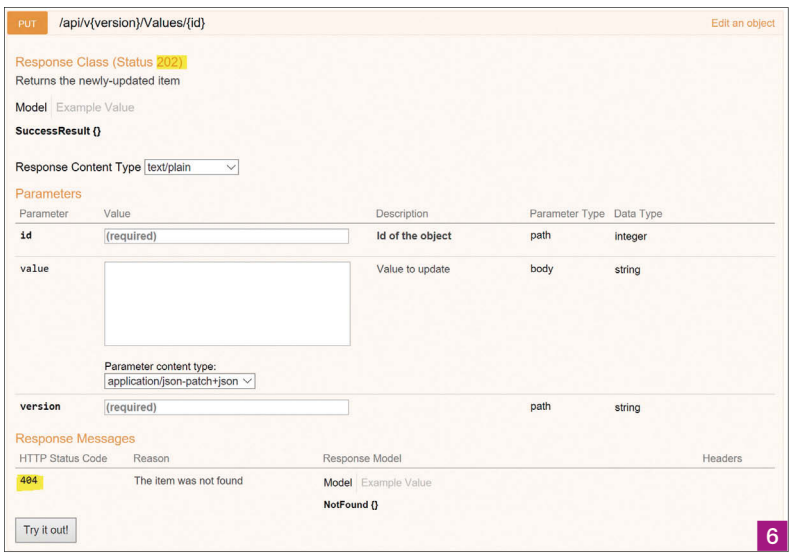
```
namespace aclerbois.swagger.Controllers.Version2
{
  [ApiVersion("2.0")]
  [ApiExplorerSettings(GroupName = "v2")]
  [Route("api/v{version:apiVersion}/[controller]")]
  public class ValuesController : Controller
  {
    // Contenu de la classe
  }
}
```

Description des types de réponse

Il est possible pour certaines actions de retourner des résultats différents par rapport au statut de réponse à la demande réalisée. Par exemple, l'action Put retourne une instance du type Success si l'objet a été édité avec succès mais si l'objet n'est pas trouvé, l'action retourne une instance du type NotFound.
Voici le code de l'implémentation de la méthode Put :

```
/// <summary>
/// Edit an object
/// </summary>
/// <param name="id">Id of the object</param>
/// <param name="value">Value to update</param>
/// <response code="202">Returns the newly-updated item</response>
/// <response code="404">The item was not found</response>
[HttpPut("{id}")]
[ProducesResponseType(typeof(SuccessResult), 202)]
[ProducesResponseType(typeof(NotFound), 404)]
public IActionResult Put(int id, [FromBody]string value)
{
  IActionResult result = null;
  // do stuff
  return result;
```

Et le résultat Swagger généré : 6



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

Coder un sine scroll sur Amiga 500

Partie 3

Cet article est le troisième 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 durant un temps. Par exemple, dans cette cracktro du groupe Supplex (Figure 1).

1 *Sine scroll dans une crakctro du groupe Supplex (https://www.youtube.com/watch?v=tlhxK4MYLuc).*

Dans le premier article, nous avons vu comment installer un environnement de développement sur un Amiga émulé avec WinUAE et coder la Copper list de base pour afficher quelque chose à l'écran. Dans le deuxième article, nous avons vu 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.

Dans ce troisième article, nous allons rentrer dans le vif du sujet en voyant comment dessiner et animer le sine scroll, d'abord au CPU, puis au Blitter.

Vous pouvez télécharger l'archive contenant le code et les données du programme sur le site de programmez.com.

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.

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...

Faire défiler et animer le sine scroll

La boucle principale peut maintenant être intégralement décrite. Elle accomplit successivement les tâches suivantes à chaque itération :

- attendre que le faisceau d'électron a terminé de dessiner la trame ;
- permuter circulairement les trois bitplanes pour afficher la dernière image ;
- attendre le Blitter et démarrer l'effacement du bitplane C contenant la pénultième image ;
- dessiner le texte dans le bitplane B qui contenait l'antépénultième image ;
- animer l'indice de la première colonne du premier caractère du texte à dessiner ;
- animer le sinus de cette première colonne ;
- tester si le bouton gauche de la souris est pressé.

Les trois premières tâches ont déjà été décrites. Nous allons donc décrire les suivantes.

Le sine scroll est dessiné par une boucle qui dessine SCROLL_DX colonnes de caractères consécutifs du texte à partir de la colonne SCROLL_X dans le bitplane. Les indices de la première colonne à dessiner et du premier caractère dont elle provient sont conservés dans les variables scrollColumn et scrollChar, respectivement. L'offset du sinus de la première colonne du sine scroll est conservé dans une variable angle.

Réglons immédiatement la question de l'animation du sine scroll dans la boucle principale.

Faire défiler le texte le long d'un sinus ne présenterait pas beaucoup d'intérêt si ce dernier n'était pas lui-même animé : il donnerait simplement l'impression de parcourir des montagnes russes. Pour cela, nous décrémentons l'offset du sinus de la première colonne du sine scroll à chaque trame, non sans oublier de gérer l'éventuel débordement :

```
  move.w angle,d0
  sub.w #(SINE_SPEED_FRAME<<1),d0
  bge _angleFrameNoLoop
  add.w #(360<<1),d0
_angleFrameNoLoop:
  move.w d0,angle
```

De plus, le texte doit défiler de gauche à droite. Pour cela, nous incrémentons de SCROLL_SPEED l'indice de la première colonne dans le texte. Ici, nous devons gérer deux éventuels débordements : le débordement d'un caractère pour passer de la dernière colonne

d'un caractère à la première du caractère suivant, et le débordement du texte pour passer du dernier caractère du texte au premier :

```
  move.w scrollColumn,d0
  addq.w #SCROLL_SPEED,d0
  cmp.b #15,d0          ;La colonne est après la dernière du caractère ?
  ble _scrollNextColumn     ;Si non, en rester là
  sub.b #15,d0          ;Si oui, colonne dans le caractère suivant...
  move.w scrollChar,d1
  addq.w #1,d1              ;...et passage au caractère suivant
  lea text,a0
  move.b (a0,d1.w),d2
  bne _scrollNextChar   ;Le caractère est après le dernier caractère ?
  clr.w d1              ;Si oui, reboucler sur le premier caractère
_scrollNextChar:
  move.w d1,scrollChar
_scrollNextColumn:
  move.w d0,scrollColumn
```

Nous pouvons maintenant passer au dessin du sine scroll. Ce dernier est assuré par la boucle évoquée plus tôt, qui est logée dans la boucle principale. Avant d'entamer cette boucle, nous devons procéder à un certain nombre d'initialisations dans l'objectif de maximiser l'utilisation des registres pour minimiser la nécessité d'aller chercher des données en mémoire.
Tout d'abord, nous déterminons l'offset (D6) du mot dans le bitplane dans lequel se trouve le bit qui correspond à la première colonne où dessiner, et nous identifions ce bit (D7) :

```
  ;Déterminer l'offset du mot du bitplane de la première colonne où dessiner

  moveq #SCROLL_X,d6
  lsr.w #3,d6    ; Offset de l'octet de la colonne dans le bitplane
  bclr #0,d6     ; Offset du mot (revient à lsr.w #4 puis lsl.w #1)

  ;Déterminer le bit de ce mot correspondant à cette colonne

  moveq #SCROLL_X,d4
  and.w #$000F,d4
  moveq #15,d7
  sub.b d4,d7   ;Bit dans le mot
```

Ensuite, nous déterminons l'adresse (A0) du caractère courant ainsi que celle (A1) du mot dans la police 16x16 de sa colonne courante (D4) à dessiner dans la colonne courante du bitplane évoquée à l'instant :

```
  move.w scrollChar,d0
  lea text,a0
  lea (a0,d0.w),a0
  move.w scrollPixel,d4
  clr.w d1
  move.b (a0)+,d1
  subi.b #$20,d1
  lsl.w #5,d1           ;32 octets par caractère dans la police 16x16
  move.w d4,d2          ;Colonne du caractère à dessiner
  lsl.w #1,d2           ;2 octets par ligne dans la police 16x16
  add.w d2,d1
  move.l font16,a1
  lea (a1,d1.w),a1      ;Adresse de la colonne à dessiner
```

Dans le code précédent, l'offset de la première colonne d'un caractère se déduit du code ASCII de ce caractère auquel il faut soustraire $20 – aspect pratique de la police 8x8 qui a servi de base à la police 16x16, les caractères sont ainsi triés.
Enfin, nous procédons à des initialisations diverses de registres utilisés de manière récurrente dans la boucle, dont l'offset du sinus de la colonne courante (D0) et le nombre de colonnes restant à dessiner (D1) :

```
  move.w angle,d0
  move.w #SCROLL_DX-1,d1
  move.l bitplaneB,a2
```

In fine, les registres se présentent ainsi au démarrage de la boucle :

Registre	Contenu
D0	Offset du sinus de la colonne courante où dessiner dans le bitplane
D1	Colonne courante où dessiner dans le bitplane
D4	Colonne courante du caractère courant à dessiner
D6	Offset du mot dans le bitplane qui contient la colonne courante où dessiner
D7	Bit dans ce mot qui correspond à cette colonne
A0	Adresse du caractère courant dans le texte
A1	Adresse du mot dans la police correspondant à la colonne courante de ce caractère
A2	Adresse du bitplane où dessiner

La boucle de dessin des SCROLL_DX colonnes du sine scroll accomplit successivement les tâches suivantes à chaque itération :
- calculer l'adresse du mot contenant le premier pixel de la colonne dans le bitplane ;
- afficher la colonne courante du caractère courant ;
- passer à la colonne suivante du caractère courant, ou à la première colonne du caractère suivant, ou à celle du premier caractère du texte ;
- décrémenter l'angle de la colonne courante.

Le calcul de l'adresse (A4) du mot contenant le premier pixel de la colonne dans le bitplane s'appuie sur une multiplication par une valeur de sinus précalculée par une puissance de 2, comme nous l'avons déjà vu :

```
  lea sinus,a6
  move.w (a6,d0.w),d1
  muls #(SCROLL_AMPLITUDE>>1),d1
  swap d1
  rol.l #2,d1
  add.w #SCROLL_Y+(SCROLL_AMPLITUDE>>1),d1
  move.w d1,d2
  lsl.w #5,d1
  lsl.w #3,d2
  add.w d2,d1   ;D1=(DISPLAY_DX>>3)*D1=40*D1=(32*D1)+(8*D1)=(2^5*D1)+(2^3*D1)
```

```
add.w d6,d1
lea (a2,d1.w),a4
```

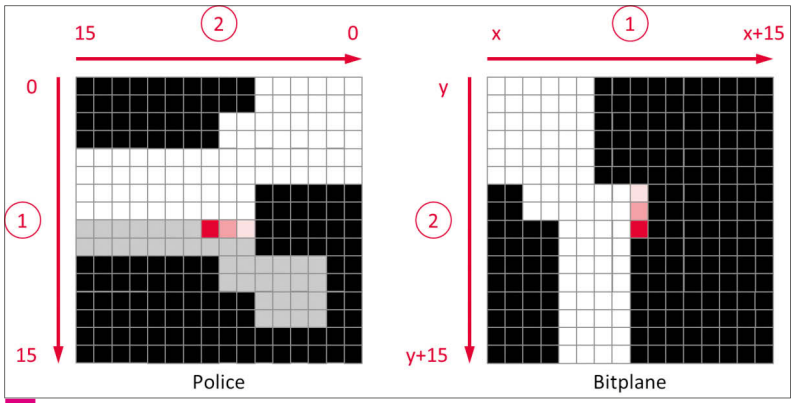
Oui ! D1 est utilisé ici comme variable temporaire alors qu'il a été initialisé pour servir de compteur à la boucle. C'est qu'ainsi que nous allons bientôt le constater, nous sommes à court de registres. En conséquence, la boucle commence et se termine par de brefs échanges avec la pile :

```
_writeLoop:
  move.w d1,-(sp)
  ;...
  move.w (sp)+,d1
  dbf d1,_writeLoop
```

Nous pouvons alors afficher la colonne courante du caractère courant dans la colonne courante du bitplane. Le principe de l'affichage est assez simple du fait que la police 16x16 a subi une rotation qui permet de tester les bits successifs d'un mot correspondant à la colonne à afficher plutôt que de tester le même bit de mots successifs : 2

Afficher la colonne courante (mot en A1) du caractère courant dans la colonne courante du bitplane (bit D7 du mot en A4), cela donne :

```
  move.w (a1),d1
  clr.w d2
  moveq #LINE_DX,d5
_columnLoop:
  move.w (a4),d3
  btst d2,d1
  beq _pixelEmpty
  bset d7,d3
  bra _pixelFilled
_pixelEmpty:
  bclr d7,d3
_pixelFilled:
  move.w d3,(a4)
  lea DISPLAY_DX>>3(a4),a4
  addq.b #1,d2
  dbf d5,_columnLoop
```



2 *Affichage d'un caractère pixel par pixel.*

Cet affichage accompli, nous pouvons passer à celui de la colonne suivante du texte, c'est-à-dire la colonne suivante du caractère courant ou la première colonne du caractère suivant, à moins que le caractère courant ne soit le dernier et que nous devions reboucler sur le premier caractère du texte, répétant le texte du sine scroll à l'infini :

```
  addq.b #1,d4
  btst #4,d4
  beq _writeKeepChar
  bclr #4,d4
  clr.w d1
  move.b (a0)+,d1
  bne _writeNoTextLoop
  lea text,a0
  move.b (a0)+,d1
_writeNoTextLoop
  subi.b #$20,d1
  lsl.w #5,d1
  move.l font16,a1
  lea (a1,d1.w),a1
  bra _writeKeepColumn
_writeKeepChar:
  lea 2(a1),a1
_writeKeepColumn:
```

Cette colonne suivante se trouvera à une autre ordonnée, déterminée en décrémentant l'offset courant du sinus… :

```
  subq.w #(SINE_SPEED_PIXEL<<1),d0
  bge _anglePixelNoLoop
  add.w #(360<<1),d0
_anglePixelNoLoop:
```

…et elle sera affichée dans la colonne suivante du bitplane, éventuellement dans le mot suivant le mot courant :

```
  subq.b #1,d7
  bge _pixelKeepWord
  addq.w #2,d6
  moveq #15,d7
_pixelKeepWord:
```

Dessiner à l'économie grâce au Blitter

L'affichage des colonnes au CPU est une tâche intensive pour ce dernier. Il existe un moyen expéditif pour le soulager : recourir au Blitter.

Dans un précédent article, nous avons vu que le Blitter permet de copier des blocs et de tracer des droites. Cette dernière fonctionnalité est particulièrement intéressante en l'espèce, car le Blitter peut tracer une ligne reproduisant un motif de 16 pixels. Ce motif ne pourrait-il pas être la colonne d'un caractère à dessiner ? Certainement. Nous allons donc utiliser le Blitter pour tracer autant de lignes de 16 pixels, verticales et texturées, qu'il y a de colonnes du sine scroll à dessiner.

Le dessin d'une colonne au CPU s'effectue de haut en bas, mais le

dessin au Blitter doit s'effectuer en sens inverse. En effet, le motif de la colonne est orienté de telle sorte que son bit 15 ne correspond pas au premier pixel de la colonne, mais à son dernier (Figure 3).

Configurer le Blitter pour tracer des lignes est un peu fastidieux, car il faut stocker des valeurs dans nombre de registres. Une bonne partie de cette initialisation peut être effectuée une fois pour toutes les lignes à tracer. En effet, le contenu des registres concernés n'est pas modifié lors d'un tracé de ligne. Il vaut donc pour toutes les lignes. Commençons par définir quelques paramètres pour nous y retrouver :

```
LINE_DX=15    ;# de lignes de la droite -1 : LINE_DX=max (abs(15-0),abs(0,0))
LINE_DY=0 ;# de colonnes de la droite -1 : LINE_DY=min (abs(15-0),abs(0,0))
LINE_OCTANT=1
```

Procédons ensuite à la partie récurrente de l'initialisation du Blitter :

```
  move.w #4*(LINE_DY-LINE_DX),BLTAMOD(a5)
  move.w #4*LINE_DY,BLTBMOD(a5)
  move.w #DISPLAY_DX>>3,BLTCMOD(a5)
  move.w #DISPLAY_DX>>3,BLTDMOD(a5)
  move.w #(4*LINE_DY)-(2*LINE_DX),BLTAPTL(a5)
  move.w #$FFFF,BLTAFWM(a5)
  move.w #$FFFF,BLTALWM(a5)
  move.w #$8000,BLTADAT(a5)
  move.w #(LINE_OCTANT<<2)!$F041,BLTCON1(a5)     ;BSH3-0=15, SIGN=1,
OVF=0, SUD/SUL/AUL=octant, SING=0, LINE=1
```

Pour chaque colonne à dessiner, nous devons fournir au Blitter l'adresse du mot contenant le pixel de départ de la droite dans BPLCPTH / BLTCPTL et BLTDPTH / BLTDPTL, le numéro de ce pixel dans BLTCON0 et le motif de la droite dans BLTADAT.

```
  WAITBLIT
  lea LINE_DX*(DISPLAY_DX>>3)(a4),a4
  move.l a4,BLTCPTH(a5)
  move.l a4,BLTDPTH(a5)
  move.w (a1),BLTBDAT(a5)
  move.w d7,d2
  ror.w #4,d2
  or.w #$0B4A,d2
  move.w d2,BLTCON0(a5)       ;ASH3-0=pixel, USEA=1, USEB=0, USEC=1, USED=1,
LF7-0=AB+AC=$4A
```

Une écriture dans BLTSIZE demandant le tracé d'une ligne de 16 pixels permet de lancer le Blitter :

```
  move.w #((LINE_DX+1)<<6)!$0002,BLTSIZE(a5)
```

Ce code vient presque se substituer à celui utilisé pour dessiner une colonne au CPU. La seule différence notable est la manière dont cette colonne est identifiée. Avec le CPU, c'est le numéro du bit dans le mot courant qui est utilisé. Avec le Blitter, c'est le numéro du pixel dans ce mot. Ces numérotations sont en sens inverse l'une de l'autre : le pixel 15 correspond au bit 0, le pixel 14, au bit 1, etc. Dans le source, les codes des versions au CPU et au Blitter coexistent. Une constante BLITTER permet de basculer de l'une à l'autre pour tester (0 pour dessiner au CPU et 1 pour dessiner au Blitter) :

```
BLITTER=1
```

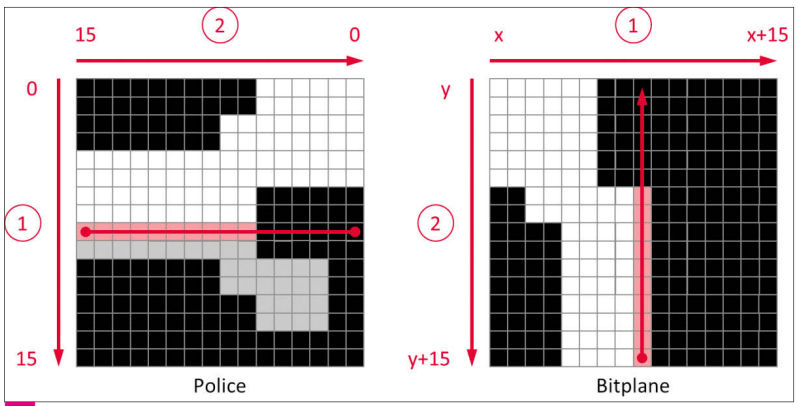
La valeur de cette constante conditionne la compilation de parties du code. Par exemple :

```
  IFNE BLITTER

  moveq #SCROLL_X,d7
  and.w #$000F,d7

  ELSE

  moveq #SCROLL_X,d4
  and.w #$000F,d4
  moveq #15,d7
  sub.b d4,d7

  ENDC
```

C'est fini ! Le sine scroll défile à l'écran. Toutefois, il reste à l'enjoliver en rajoutant quelques effets peu coûteux en cycles, et surtout à en optimiser le code…

Liens utiles

WinUAE : http://www.winuae.net/
Amiga Forever : https://www.amigaforever.com/
ASM-One : http://www.theflamearrows.info/documents/ftp.html
ReqTools : http://aminet.net/package/util/libs/ReqToolsUsr.lha
Amiga Hardware Reference Manual :
http://amigadev.elowar.com/read/ADCD_2.1/Hardware_Manual_guide/node0000.html
M68000 8-/16-/32-Bit Microprocessors User's Manual :
http://www.nxp.com/assets/documents/data/en/reference-manuals/M68000PRM.pdf
M68000 Family Programmer's User Manual :
http://cache.freescale.com/files/32bit/doc/ref_manual/MC68000UM.pdf
Le manuel d'ASM-One : https://archive.org/details/AsmOne1.02Manual



3 *Affichage d'une colonne d'un caractère au Blitter.*

le code est toujours plus beau dans le repo du voisin

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 : C. Villeneuve, A. Varlenglen, B. Cornet, D. Carteau, L. Daval, M. Amokhtarl, D. Duplan, A. Clerbois, G. Giacomini, C. Michaud, P. Lorieul, D. Volturon, Mika, A. Boudou, V. Perrin
Couverture : © Petmal - **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, février 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

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,
HTML 5 / CSS 3, ...

Java · Microsoft .NET · node JS · Kotlin · Swift · JS · php · HTML 5 · CSS 3

FAIRE EST LA MEILLEURE FAÇON DE PENSER

CACD2

La manufacture digitale du groupe Crédit Agricole

Notre mission est de concevoir, développer
et industrialiser des services et des produits digitaux
pour toutes les entités du groupe, autour de
problématiques fondamentales comme la finance,
l'immobilier, la santé ou encore la sécurité.

POUR PLUS D'INFORMATIONS, CONTACTEZ-NOUS :

recrutement@cacd2.fr ou cacd2.fr


WINDEV
23
PLATEFORME INTÉGRÉE
DE DÉVELOPPEMENT
WINDEV®
DÉVELOPPEZ 10 FOIS PLUS VITE
PCSOFT