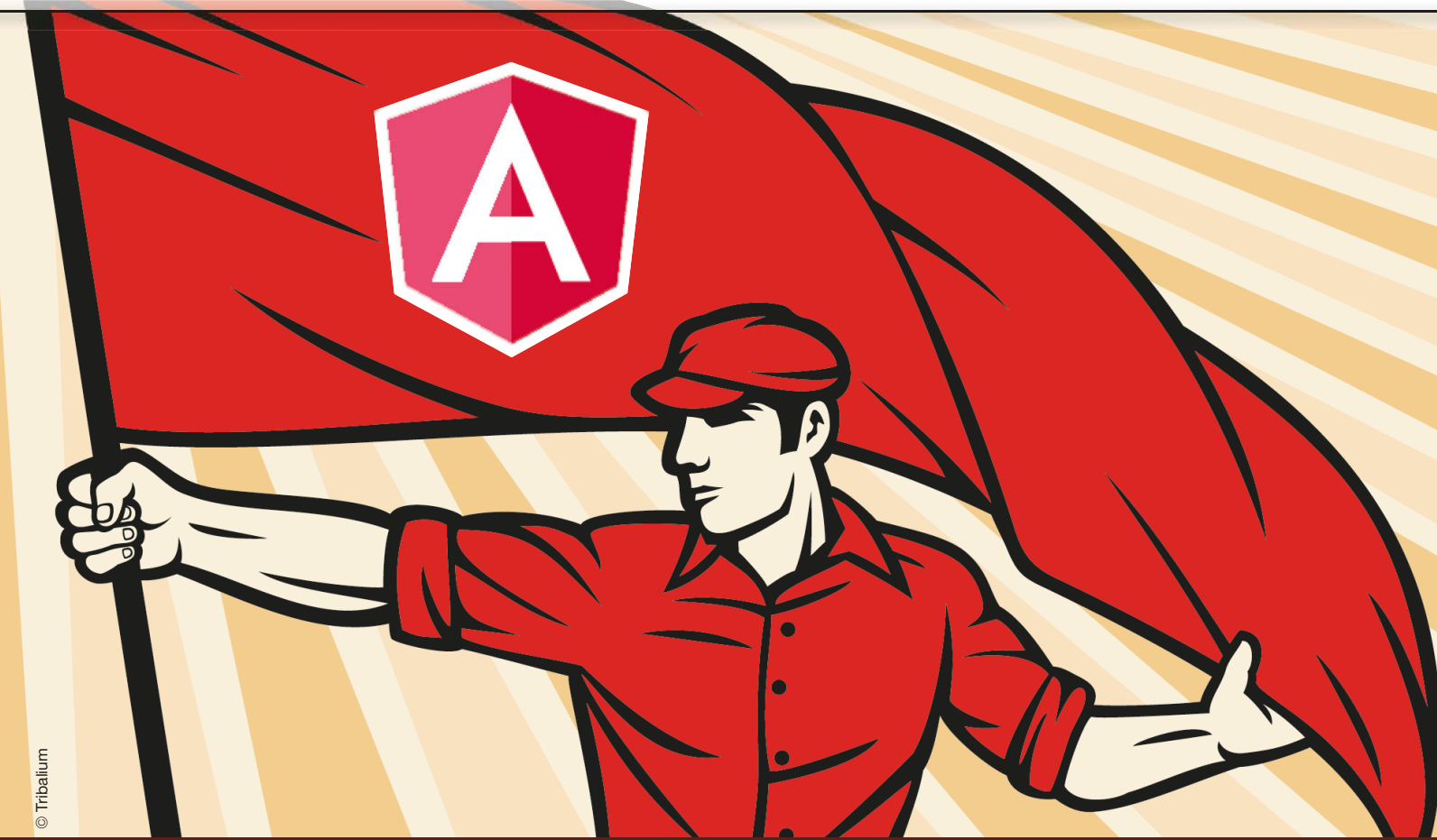
PROGRAMMEZ!

le magazine des développeurs

#203 - janvier 2017



LA NOUVELLE REVOLUTION DU WEB

ANGULAR
Angular fait sa révolution

LES BOTS
Développer son bot en quelques minutes

WEBGL
La 3D native sur les navigateurs

PROGRESSIVE WEB APP
Le web mobile devient vraiment mobile

LARAVEL
Le framework PHP qui va vous plaire

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

M 04319 - 203S - F: 6,50 € - RD

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

CLOUD | INFOGÉRANCE | SERVEUR DÉDIÉ | VPS | MESSAGERIE

“pure énergie”(1)

Pour bien démarrer 2017, la pure énergie ne sera pas de trop. Cette nouvelle année promet beaucoup de choses : des nouveautés et beaucoup de continuité. Tout d'abord, sur la partie plateformes mobiles, il ne faut pas s'attendre à beaucoup de mouvements. Android et iOS continueront à dominer. Hormis une très très grosse surprise, Windows Mobile continue sur sa lancée actuelle : aucune existence. Pour Microsoft, la question va se poser : continuer ou arrêter.

Cependant, Microsoft a posé des jalons stratégiques pour 2017 et après : la mise à jour de Windows 10 promet beaucoup pour la 3D, la réalité virtuelle. Plus que jamais l'open source sera aussi une des priorités. 2017 devrait confirmer le renouveau de l'éditeur sur plusieurs marchés technologiques. Reste à réellement décoller sur la partie IoT / Maker. Sur ce marché IoT, Google a enfin décidé de bouger avec Android Things après des mois de silence autour du projet Brillo. Par contre, Google patine sur la partie réalité augmentée / virtuelle. Après l'arrêt des lunettes v1, on attend une possible v2 et surtout le projet Magic Leap dans lequel, Google a investi, mais qui serait bien moins avancé que les démos voulaient nous le faire croire.

Côté Java et Oracle, c'est toujours compliqué. Java 9 devrait arriver au milieu de l'été, si tout va bien. Mais l'éditeur ne rassure pas réellement sur l'avenir de la plateforme et du langage. Et sur MySQL, on attend toujours des signes forts de l'implication de l'éditeur. Et si finalement Oracle se délestait de Java et de MySQL au profit d'une fondation telle que Apache, ou une autre ?

Côté Apple, il faut bien en parler de temps en temps, on attend toujours le renouvellement complet des gammes Mac qui n'ont pas brillé en 2016. Plusieurs problèmes peuvent expliquer ces retards : une architecture Intel en retard, des problèmes sur la partie GPU, pourquoi pas une première machine ARM dès 2017, l'apparition de nouvelles technologies comme la réalité virtuelle / mixte. L'autre question qui se pose : que va sortir Apple pour le 10e anniversaire de l'iPhone ?

Et pour vous, 2017 sera :

- L'espoir d'avoir un meilleur salaire 26 % ;
- Se former à de nouvelles technologies 21 % ;
- Ne plus développer dans l'urgence 7 % ;
- Changer de poste 8 % ;
- Etre écouté quand vous proposez des choix techniques 5 % ;
- Obi-wan Kenobi : un classique indémodable 34 %.

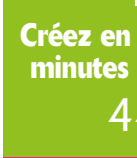
(sondage express Programmez! de décembre 2016)

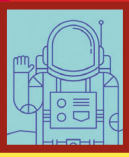
Pour débuter cette nouvelle année, nous vous proposons un dossier gargantuesque : la nouvelle révolution du Web. Et comme vous le verrez, le Web connaît de profondes évolutions qui impactent directement le développeur.

Bonne année 2017

ftonic@programmez.com

(1) Référence à une des phrases cultes de M. Spock





Dans le prochain numéro !
Programmez! #204, dès le 3 février 2017

Microsoft + Open Source = du rejet à la réalité

Comment Microsoft et l'open source se sont trouvés après 10 ans de débats, d'affrontements et de rapprochements ? Aujourd'hui, Microsoft supporte les outils open source et ouvre de nombreux projets et codes. A la plus grande joie des développeurs ?

Choisir son environnement de développement mobile

Natif, Web, hybride ? Mono-plateforme ou multiplateforme ? Le développement mobile a beaucoup évolué en 10 ans. Quelle solution choisir ?
Quel type de développement faire ? Programmez! vous aide à bien choisir !

MSCloudSummit

Paris / 23 au 25 janvier

La Conférence MS Cloud Summit se déroulera du 23 au 25 janvier 2017 au NewCap Event Center, Paris 15e. Il s'agit de la plus importante conférence communautaire en France en 2017 : plus de 600 participants professionnels de l'environnement Microsoft sont attendus ! L'agenda comprend 72 sessions. Le 25 janvier, Julia White, Microsoft Corporate Vice-President Azure et Marketing Sécurité nous présentera les directions du Cloud Microsoft. Pour vous inscrire, une participation aux frais de 15 vous sera demandée :

- 24 janvier, orientée Plateforme de Données ;
- 25 janvier, orientée Cloud.

Les meilleurs speakers français et internationaux seront présents. Une occasion unique pour discuter Cloud. Programmez! est partenaire de l'événement.
Inscription & informations sur : https://mscloudsummit.fr/fr/accueil/

Quelques sorties SF attendues

Rogue One : Ok c'est fait ! Et c'est plutôt bon.

La momie : 9 juin 2017. Remake d'un des grands classiques du cinéma. La Momie de Stephen Sommers était tellement sympa.

Alien : Convenant : 19 mai 2017 ! Nous avions beaucoup aimé Prometheus (pas le meilleur de Ridley mais tout de même). Un seul mot : courez !

Blade Runner 2049 : attendu pour octobre 2017 si tout va bien. Alors des réponses à nos questions ?

Star Wars épisode 8 : 15 décembre 2017

Stargate : ce remake du classique Stargate ne se ferait finalement pas.

Matrix : nouvelle trilogie ou pas ? Cette rumeur revient régulièrement depuis 2 ans.

Pour se faire plaisir :

Twin Peaks saison 3: toujours annoncé pour 2017 mais sans date précise, David Lynch fidèle à lui-même !

Game of Thrones saison 7 : été 2017. Impatient ?

Star Trek Discovery : nouvelle série Star Trek, distorsion annoncée pour mai.

Kaamelott : le tournage du 1er film devrait débuter en janvier ou février 2017. Alors, heureux ?

Quoi de neuf avec NetBeans ?

Après les multiples retards de Java 9, les doutes sur JEE, NetBeans n'était pas forcément en meilleure santé. La version 8.2 a été sortie début octobre 2016.
Désormais, la roadmap officielle prévoit :

- NetBeans 9.0 au milieu de l'été ;
- NetBeans 9.1 dès la fin septembre, début octobre.

Bref, mieux vaut attendre la 9.1.

Microsoft

se félicite du mauvais accueil (relatif) des nouveaux MacBook Pro qui aideraient à vendre des Surface.

Super Mario

est plus fort que Pokémon Go

Java 9

est attendue pour le 27 juillet… Oui, non, peut-être ?

Visual Studio 2017 RC

a été mise à jour le 12 décembre dernier : bug fix et diverses améliorations, dixit l'équipe VS.

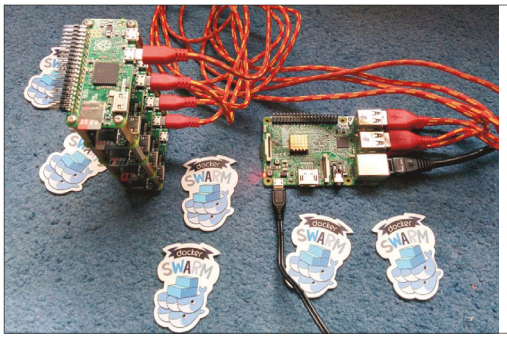
AGENDA

Quelques rendez-vous des communautés

- Paris Jug :
  10 janvier : soirée Young Blood VI
- CocoaHeads Paris :
  12 janvier : backend CloudKit. Conférence chez Bla Bla Car
- Meetup .Net Toulouse :
  16 janvier 2017 : TypeScript 2, le futur de JS et les transpileurs

GOOGLE MISE SUR ANDROID THINGS POUR LES IOT

Souvenez-vous, Google avait présenté le système pour IoT, Brillo et le protocole Weave mais depuis l'annonce, Brillo ne semblait pas beaucoup avancer et Google restait plutôt discret sur le sujet. Finalement, Brillo change de nom et devient Android Things, ce qui est cohérent avec le branding des marques autour d'Android. Le projet est accessible en préversion développeur mais Google a clairement pris du retard sur la mise en place de ce nouveau système dédié IoT. Actuellement, vous pouvez déployer sur Intel Edison / Joule, NXP Pico et Raspberry Pi 3. Le SDK se veut proche de l'expérience développeur d'Android pour ne pas trop dérouter les développeurs.
Android Things permet d'avoir un écosystème Android connu, des outils connus, avec des API et librairies dédiées IoT. Il se pose en alternative forte à un Windows 10 IoT par exemple. Google, avec ce projet, porte la un coup à Microsoft, même si ce dernier a une avance mais a du mal à se positionner et à le promotionner. Par contre, on retrouve le caractère verbeux, pour nous, du langage qui allourdit le développement IoT.
Site : https://developer.android.com/things/index.html



Tu as des

Rapsberry Pi Zero

et tu ne sais pas quoi en faire ? Monte un cluster Docker Swarm, c'est fun ! site :
http://blog.alexellis.io/live-deep-dive-pi-swarm/

[2017]

WINDEV 22

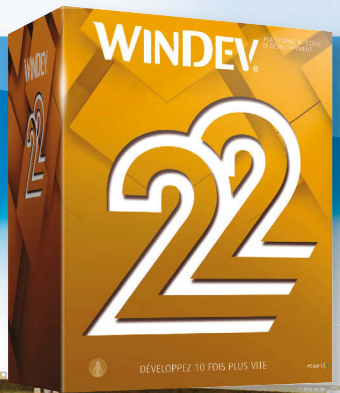
922 NOUVEAUTÉS TECHNOLOGIQUES INDISPENSABLES

La nouvelle version 22 de WINDEV est la version **la plus riche** en nouveautés.

Parmi les 922 nouveautés vous bénéficierez de nouvelles technologies indispensables : ● **Le nouveau champ Traitement de Texte** permet de créer et manipuler des documents **sans sortir de l'application** (et également de les gérer en WLangage) ● Le nouvel **éditeur d'images** orienté développeur permet de retoucher vos images, créer vos icones et vos images «5 états» **sans quitter l'environnement** ● 22 nouveautés boostent les **tables** ● 11 nouveautés boostent vos **plannings** ● **Les nouveaux graphes** ● **IOT** (Objets connectés): support de la norme MQTT ● Pour améliorer la **vitesse des requêtes, HFSQL** trouve les meilleures clés d'index de chaque serveur en exploitation ● **HFSQL** bénéficie d'un **nouveau tableau de bord** ● Installez votre **GDS dans le cloud** en 3 clics pour **2 Euros par mois** ● **Nouveau GDS visuel:** gérez les branches d'un clic ● Les centaines de nouvelles fonctions **WLangage** ● **Le Code Coverage** affiche le pourcentage de code testé ● Le **GO** de projet WINDEV Mobile dans WINDEV ● Dans **WINDEV Mobile 22**, vous êtes averti immédiatement si une ligne de code que vous tapez ne fonctionnera pas sur un système ● Depuis WEBDEV, utilisez des composants **Angular JS**, **Bootstrap**, **jQuery UI**... ● **Télémétrie sur mobile** ● Créez des **Webservices REST** ● Créez des sites complets dans une seule page (**Single Page Application**) ● Utilisez les mots de passe de **Facebook**,... comme identifiants de vos applications et vos sites ● **WebSocket**: c'est le serveur Web qui envoie lui-même les données modifiées aux pages ● Intégrez automatiquement les **trackers Google Analytics** dans vos sites ● Etc, Etc... (liste exhaustive des nouveautés dans la revue de 92 pages accessible sur pcsoft.fr)

Tél Paris: 01 48 01 48 88
Tél Montpellier: 04 67 032 032

WWW.PCSOFT.FR



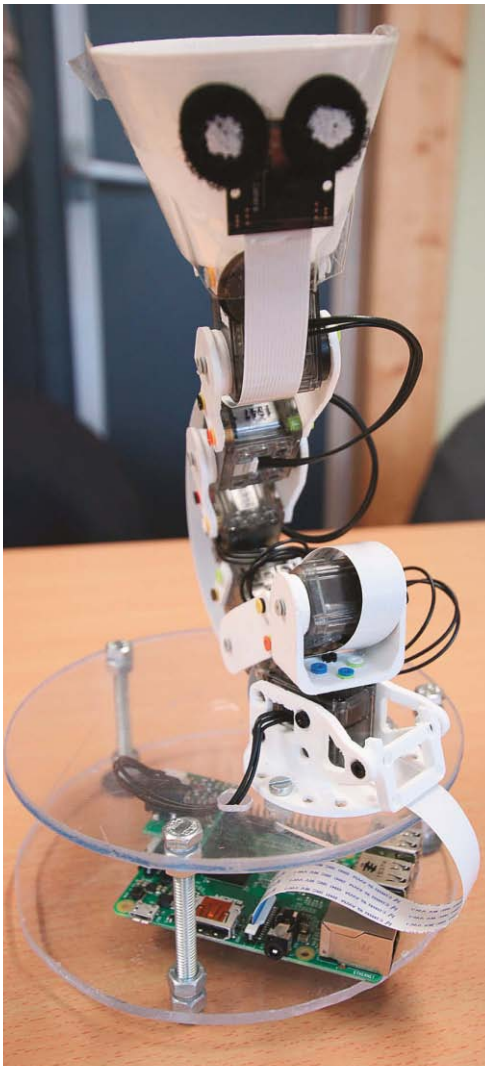
AFFLUENCE RECORD DANS LES 12 VILLES DU «WINDEV TOUR 22»

Merci de votre fidélité

Poppy Ergo Jr : un kit robotique pour l'enseignement des sciences du numérique

Poppy Ergo Jr est un robot open-source, conçu pour être utilisé facilement en classe pour initier aux sciences du numérique, notamment à l'informatique et à la robotique.

• Stéphanie Noirpoudre,
Ingénieure pédagogique dans l'équipe **Flowers.**



Les pieces sont imprimables en 3D. L'utilisation de rivets rend les modifications simples à réaliser. Il est donc possible de le monter soi-même et de le programmer : https://www.youtube.com/watch?v=T1ZxHZ0POk0

Le kit Poppy Ergo Jr propose une programmation visuelle avec Snap! (similaire à Scratch). Avec Snap!, les apprenants sont amenés à assembler des blocs d'instructions et à les activer, de façon très intuitive (grâce aux codes couleurs et aux formes), pour voir directement sur le robot Ergo Jr les effets des programmes qu'ils construisent.

Pour les plus experts, il est aussi possible de le programmer en Python ou avec les langages de votre choix grâce à une API REST qui permet d'envoyer des commandes et de recevoir des informations provenant du robot avec de simples requêtes HTTP.

Un ensemble de ressources et d'activités

Le robot Poppy Ergo Jr a ete cree dans le cadre du projet Poppy Education de l'Equipe de recherche FLOWERS (Inria, ENSTA Paris Tech), soutenu par la Region Nouvelle-Aquitaine et les Fonds Europeens FEDER. Poppy Education vise à développer, évaluer et distribuer des kits robotiques clés en mains à visée éducative, afin de promouvoir la robotique et le numérique dans l'éducation et de faciliter l'acquisition de connaissances en informatique.

Il est accompagné d'un ensemble d'activités pédagogiques gratuites mises au point avec des enseignants du secondaire, allant de la découverte du robot à une utilisation plus complexe.

Ces activités sont regroupées dans le livret pédagogique Apprendre à programmer Poppy Ergo Jr en Snap! et sont également disponibles sur le forum du projet Poppy où chacun est invité à les commenter et à en créer des nouvelles.

Le robot Ergo Jr et les activités proposées sont testés et évalués sur le terrain grâce à la collaboration d'enseignants de collèges et de lycées de la région Aquitaine.

Voici des exemples de projets réalisés :

- Poppy Ergo Jr joue au chamboule-tout : une première séance pour prendre en main son robot (seconde ICN). Voir l'activité en vidéo et sa description
- Contrôler la tête de l'Ergo à la souris : une séance de découverte (seconde ICN - Lycée François Mauriac). Vidéo et Présentation de la séance
- Une partie de Tic Tac Toe - Humain VS Ergo Jr (Projet terminal ISN - Lycée Camille Jullian). Voir le projet résultat en vidéo et le journal de bord des élèves
- Modélisation du port automatisé de Rotterdam (Projet mené en technologie en 3e au collège Anatole France à Cadillac). Voir la vidéo et le retour d'expérience par l'enseignant

- Atelier robotique au CERN - une journée pour construire et programmer son robot. Pour revivre l'atelier en images, voici une vidéo créée par un des participants, Guillaume Delille (étudiant de terminale au lycée Jeanne d'Arc de Gex)

Si, vous aussi, vous voulez réaliser des activités en classe ou chez vous, n'hésitez pas à utiliser les ressources ci-dessous :

- Obtenez votre robot : Le kit Ergo Jr est distribué par Génération Robots. Vous pouvez également utiliser les fichiers source pour le construire par vos propres moyens.
- Construisez le robot Poppy Ergo Jr : documentation du montage
- Echangez avec la communauté Poppy grâce à son forum
- S'initier en Snap! (langage visuel par bloc) : téléchargez le livret pédagogique Programmer Poppy Ergo Jr en Snap
- Programmer en Python : tutoriel Python notebook pour commencer avec Ergo Jr •

DOMAINES | MAIL | HÉBERGEMENT | E-COMMERCE | **SERVEURS**

1&1 SERVEUR VIRTUEL CLOUD

à partir de **4,99** € HT/mois (5,99 € TTC)*

intel XEON inside

Trusted Performance.
Intel® Xeon® processors.

NOUVEAU

Avec 1&1, fini l'effet « voisin bruyant » : le serveur virtuel Cloud vous appartient à 100 % ! La nouvelle technologie Cloud est idéale pour débuter avec un serveur Web ou mail et convient aussi parfaitement aux projets exigeants, comme les applications de bases de données.

- Ressources dédiées avec la virtualisation VMware®
- Accès root complet
- Stockage SSD
- Trafic illimité
- Haute performance
- Sécurité maximale
- Meilleur rapport qualité/prix
- Assistance 24/7
- Linux ou Windows au choix
- Plesk ONYX

1 CHOIX VOTRE DURÉE D'ENGAGEMENT

1 APPEL UN EXPERT VOUS RÉPOND

1 CERTITUDE LA PROTECTION CONTRE LES DÉFAILLANCES

0970 808 911
(appel non surtaxé)

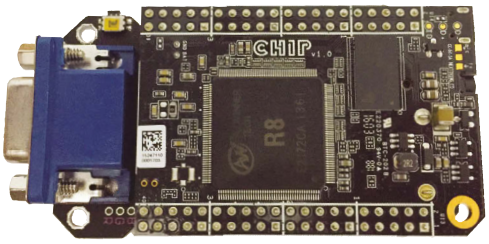
1&1

1and1.fr

*1&1 Serveur Virtuel Cloud S est à 4,99 € HT/mois (5,99 € TTC). Pas de frais de mise en service pour un engagement minimum de 12 mois. Conditions détaillées sur 1and1.fr. 1&1 Internet SARL, RCS Sarreguemines B 431 303 775.

Enfin ! Oui ! Enfin, la carte C.H.I.P. vendue 9 $ est là !

• François Tonic



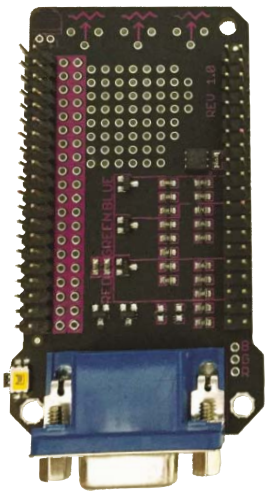
Franchement, je n'y croyais plus. La C.H.I.P. a été une des vedettes de Kickstarter : 2 millions $, 39 560 contributeurs. La promesse : une carte électronique complète pour 9 $, une taille réduite et de nombreuses extensions. Désormais produite en masse, la C.H.I.P. va pouvoir faire parler d'elle dans la communauté.

Il est facile de vouloir la comparer à une Raspberry Pi, parfois, on l'oppose à la Pi Zero, ce qui n'est pas forcément judicieux. Mais faisons donc ce comparatif : voir tableau. Chaque carte a des avantages et des inconvénients. Mais soyons honnête, les caractéristiques techniques de la C.H.I.P. ne rivalisent pas en tous points avec une Pi 3. Cela ne l'empêche pas de disposer d'atouts non négligeables : la taille, le GPIO, le stockage interne, la connectivité sans fil. Oui il n'y a pas d'Ethernet, un seul port USB, pas de HDMI, cependant, cette nouvelle carte suffit pour de nombreux usages. La disponibilité en volume a réellement débuté en novembre et en France, elle est difficile à trouver. La communauté n'est pas encore très visible, mais elle devrait rapidement croître, et plusieurs extensions sont disponibles, notamment VGA et HDMI. Il faut rajouter les extensions, l'alimentation externe, le clavier, la souris, etc., la C.H.I.P. est plutôt bien équipée. Et on en a pour notre argent !

Après le premier boot

Lancer pour la première fois cette carte n'a rien de sorcier : on prend le shield HDMI ou VGA, on branche les câbles et l'alimentation. C'est tout. L'unique port USB sera très limitant, mieux vaut avoir un hub, attention tout de même à ne pas saturer le port. La connectivité WiFi ne pose pas de souci. Le système préinstallé est un Linux pour architecture ARM. On peut voir à espérer pouvoir installer d'autres OS que celui par défaut. Pour flasher, vous pouvez soit passer en ligne de commande, soit via l'application CHIP Flasher sous Chrome. Cette partie logicielle n'est pas très « friendly ». Il est possible d'installer Docker et de l'utiliser comme barebone de conteneurs, mais attention ne soyez pas trop exigeant sur les performances. À peine la CHIP disponible que la version pro arrive. Elle est annoncée à 16 $ et propose un ARMv7 1Ghz, le tout avec le SoC GR8, 256 ou 512 Mo de mémoire, WiFi + Bluetooth, entièrement open hardware. À la différence de la CHIP, les headers ne sont pas soudés par défaut et se destine plus aux IoT. Un kit de développement est proposé, il s'agit d'une carte mère sur laquelle on pose la pro. Elle peut accueillir 2 boards.

On devrait disposer d'une meilleure base pour créer un barebone de base, une box domotique, etc. Il faudra tout de même que la partie documentation soit renforcée et que la plateforme logicielle s'améliore et soit mieux intégrée. Nous reviendrons très prochainement sur cette carte prometteuse. •



LES +

- Le prix
- Les dimensions
- Une alternative aux cartes actuelles
- Connectivité sans fil
- Stockage interne

LES –

- 1 seul port USB
- Communauté
- Disponibilité en France
- Pas de HDMI par défaut
- Pas d'Ethernet
- Modèle logiciel

	C.H.I.P.	Raspberry Pi 3	Raspberry Pi Zero
CPU	1 Ghz ARM A13	1,2 GHz ARMv8	1 Ghz ARM
GPU	ARM Mali-400	VideoCore IV	VideoCore IV
Mémoire vive	512 Mo	1 Go	512 Mo
Stockage interne	4 Go	0	0
GPIO	80	40	40
WiFi / Bluetooth	Oui/oui	Oui/oui	Non/Non
Vidéo	Composite	HDMI	HDMI
Système	Linux	Linux	Linux
Dimension (en mm)	65x30x5	56,5x85,6	40x60x13
Poids (environ)	31 g	45 g	9 g
Headers soudés	oui	oui	non
Ethernet	non	oui	non
Connexion stockage externe	USB	SD USB	SD USB
Disponibilité en France	non	oui	très difficile à trouver
Communauté	?	+++	++
Tarif officiel	9 $	35 $	5 $

UNE NOUVELLE CARTE X86 : LATTEPANDA

Le monde des cartes x86 commence lentement à s'animer. Après la JaguarBoard, c'est aujourd'hui la LattePanda qui arrive, en plusieurs éditions : 2 Go de mémoire et 32 Go de stockage ou 4 / 64. Une version arrive avec Windows 10 Home préconfigurée, théoriquement, Windows 10 IoT pourrait fonctionner ainsi que Linux. La carte est basée sur un Atom X5, une puce graphique Intel HD Graphics, 1 port USB 3 et 2 USB 2, WiFi, Bluetooth, un coprocesseur Atmega32u4 (Arduino), port HDMI, Ethernet, des GPIO compatibles Arduino (Leonardo), 6 connecteurs pour des capteurs (type Gravity). Nous aimons beaucoup les connexions pour les capteurs, la compatibilité Arduino par défaut. L'ensemble semble bien fini et de qualité. Comme souvent avec les cartes x86, le tarif peut refroidir : à partir de 89 $. Cependant, les petits plus cités plus haut apportent une vraie ouverture à la carte, contrairement à la JaguarBoard. Elle devrait être trouvable en France très rapidement. À noter qu'un kit complet de capteurs Gravity est proposé par DFRobot au prix de 59,90 $. Un écran 7" est disponible, à 34 $.

Abonnez-vous à programmez!

le magazine des développeurs

Nos classiques

1 an — 11 numéros — **49€***

2 ans — 22 numéros — **79€***

Etudiant — 1 an - 11 numéros — **39€***

* Tarifs France métropolitaine

Abonnement numérique

PDF — 1 an - 11 numéros — **35€***

Souscription uniquement sur
www.programmez.com

Option :
accès aux archives 10€

Nos offres spéciales 2017

1 an offre 2017 — 65€*

11 numéros + clé USB***
+ 1 livre numérique (au choix)
+ 4 n° vintage****

** Au lieu de 126,42 € (1 an : 49 €, clé USB : 34,99 €, livre numérique : 22,43 €, n° vintage : 20 €) / Limitée à France métropolitaine*

2 ans offre 2017 — 95€**

22 numéros + clé USB***
+ 1 livre numérique (au choix)
+ 4 n° vintage****

***Au lieu de 156,42 € (2 ans : 79 €, clé USB : 34,99 €, livre numérique : 22,43 €, n° vintage : 20 €) / Limitée à France métropolitaine*

**** Clé USB Programmez! 4 Go contenant tous les numéros depuis le n°100*
***** Selon les stocks disponibles. Antérieur au N°168*

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

☐ *Abonnement 1 an au magazine :* 65 €
11 numéros + clé USB
+ 1 livre numérique (au choix)
☐ **Scratch** ou ☐ **Arduino**
+ 4 n° vintage

☐ *Abonnement 2 ans au magazine :* 95 €
22 numéros + clé USB
+ 1 livre numérique (au choix)
☐ **Scratch** ou ☐ **Arduino**
+ 4 n° vintage

☐ M. ☐ Mme 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*

PROG 203
Offre limitée, valable jusqu'au 2 février 2017

Histoire & futur de l'informatique

Épisode 1

• Sylvain Abélard
(@abelar_s)
Architecte logiciel
Faveod

L'ordinateur est un outil formidable. Les principes de l'innovation sont toujours les mêmes : une société, une économie se posent des problèmes. On imagine, applique, croise et assemble des siècles d'idées et techniques.

Libéré de quelques contraintes, de nouveaux besoins apparaissent et on recombine ces outils, encore plus loin, toujours plus vite. L'Histoire se répète. Elle a façonné l'état de l'art actuel, elle permet d'anticiper le futur. Sous quelles conditions ? À quel coût ? Bienvenue dans cette première partie : le passé lointain des sciences et techniques, de la numérisation et de l'automatisation.

AVANT-HIER
UNE RICHE HISTOIRE
L'âge de la machine

Qu'est-ce que le "premier" ordinateur ? Difficile à dire. Mais comment a-t-on commencé à compter, calculer, prévoir, automatiser ? Quand les outils sont rudimentaires (pierre, bois, os, métal, eau), il a fallu à nos ancêtres une bonne dose d'intellect et de science ! La vie est centrée sur les récoltes : quand faut-il semer, irriguer, récolter, quelle part garder pour l'hiver et quelle part pour les impôts ? L'innovation vient de partout, les inventeurs cherchent à rentabiliser leur technique chez l'utilisateur directement, ou auprès d'un mécène. Au fur et à mesure des avancées techniques et scientifiques, les progrès combinés donnent des résultats exceptionnels, pour une croissance et une accélération exponentielles, qui ouvrent le champ des possibles et créent à leur tour de nouveaux besoins.

Calendriers

On retrouve dans la plupart des civilisations antiques des calendriers. Ce sont évidemment les moyens de lire et noter les saisons, savoir quand planter, récolter, et parfois mesurer la production et calculer les impôts. Difficile de juger des fonctions religieuses et politiques de leur temps, de l'Égypte antique à Stonehenge, des Sumériens aux Mayas, des cônes d'or rituels de l'âge du Bronze au mécanisme d'Anticythère, force est de constater que les mouvements des étoiles et les éclipses nous fascinent, et sont à la fois assez stables et assez "simples" (en termes d'opérations) à calculer et prédire.

Mécanisation, automatisation, innovation

On retrouve aussi partout des mécanismes pour compter, comme les abaques, et de nombreuses machines pour simplifier le travail. Signe des temps, on retrouve des machines pour l'irrigation, des horloges et des automates chez Ismaïl Al-Jazari et Léonard de Vinci.
Bien plus tard, l'industrie textile est un poids majeur dans l'économie des pays, et c'est tout naturellement que l'on cherche à automatiser le métier à tisser : Basile Bouchon puis Joseph Marie Jacquart relèveront le défi, à l'aide d'instructions renseignées sur des cartes perforées.
Cette idée de la carte perforée (ou de cylindre avec des pointes) fera son chemin à part : les automates de Jacques Vaucanson, les boîtes à musique, les orgues de barbarie, pianos mécaniques... Elle sera réutilisée avec bonheur jusqu'à l'arrivée du support en bandes magnétiques.
Et chacun de ces inventeurs est aussi connu pour tout un portefeuille de réalisations, parfois pratiques, parfois expérimentales, qui pourraient même sembler inutiles dans un premier temps. On croit voir dans cette histoire très riche un thème commun avec l'innovation, qui commence souvent par reprendre ce qui existe déjà dans l'état de l'art, pour en améliorer tout ou partie, reprendre et croiser les idées, puis livrer quelque chose que l'on peut qualifier de nouveau, ou dont l'usage ou la combinaison apporte de nouvelles possibilités.

Machines à calculer

Pascal fait en 1645 la démonstration de sa célèbre "machine d'arithmétique", dont l'histoire se rappellera sous le nom de Pascaline. Il ouvre ainsi la voie aux machines mécaniques à compter : c'est possible, c'est viable, et le besoin est créé. En parallèle, Leibniz invente en 1671 un cylindre cannelé qui sera utilisé dans une dizaine de machines de 1684 à 1948.
Charles Babbage inventera deux machines, la machine différentielle (calcul de polynômes, projet lancé en 1822 puis abandonné) puis la machine analytique (stockage d'instructions, mémoire, boucles, branches d'exécution différentes, 1837). Ada Lovelace sera la première (et la seule ?) à bien vouloir documenter le fonctionnement de la machine et les manières de l'utiliser, lui valant le titre de première programmeuse de l'histoire. Les idées de Pascal, plus simples et moins chères, seront reprises dans des calculatrices commercialisées sous les doux noms d'Arithmomètre (1857), Comptomètre (1887)... tandis que les idées de Babbage et Lovelace seront reprises plus tard, lors de la seconde guerre mondiale.

Human Computers

Mais on n'était pas perdus avant l'arrivée de l'ordinateur ! Les "ordinateurs humains" comptaient, classifiaient : Alexis-Claude Clairaut a même divisé le calcul, créant ainsi la première "ferme de calcul distribué" pour prédire le retour de la comète de Halley. Les "Harvard Computers", un groupe de parfois plus d'une douzaine de femmes de l'Observatoire de l'université Harvard, ont assemblé et analysé les données du spectre lumineux de nombreuses étoiles, permettant ensuite de les classifier. Tables mathématiques, almanachs nautiques, et même les calculs pour déchiffrer les messages chiffrés par Enigma (les "Top Secret Rosies")... l'algorithme (étapes de recette déterministes pour arriver à un résultat) date de bien avant l'ordinateur moderne, mais c'est l'ordinateur moderne qui lui donne ses lettres de noblesse.

L'âge de l'électron

Les progrès dans l'industrialisation et les transports vont lever de nouveaux besoins de suivi, des volumes sans précédent, une concentration économique également sans précédent.
La course à l'armement qui précède la Seconde Guerre Mondiale lance la machine, et son arrivée va précipiter tout cela dans une économie de guerre, et les cerveaux du monde entier vont se consacrer à de nouveaux problèmes : communiquer, chiffrer, décrypter bien sûr, mais aussi, puisque tout est désormais transformé en chiffres, traiter en masse des volumes de données numériques.

Systèmes de guidage

L'utilité d'une arme à projectile (obus, torpille) nécessite évidemment de sa vitesse, sa puissance... encore faut-il que la charge touche sa cible.
Dans la course aux calculs de trajectoire toujours plus précis et rapides, et grâce à la miniaturisation relative des composants nécessaires, on aura dès 1938 des calculateurs électromécaniques pour guider les torpilles.

Turing

Alan Turing est le plus connu dans cette révolution scientifique. Mathématiciens, cryptanalystes, mais aussi linguistes, champions d'échecs, de mots croisés, sont recrutés par le Bletchley Park, ainsi qu'un expert en déchiffrage de papyrus et tout un service de "human computers", dans un effort pour casser les codes d'Enigma, l'outil de chiffrement des nazis. Turing va poser les bases non seulement de l'analyse statistique pour casser les codes, des machines physiques (mécaniques ou numériques), mais aussi l'expérience de pensée qui deviendra célèbre sous le nom de "machine de Turing" : un ruban infini de papier (mémoire dans laquelle on stocke instructions et données) et une tête de lecture/écriture.

Kondrad Zuse

Beaucoup d'innovations ont été "co-découvertes" durant la guerre. Malgré l'absence de communication sur les technologies secrètes, l'état de l'art juste avant la guerre était le même partout, et les expositions universelles partageaient sinon les secrets de fabrication, le besoin, l'envie, le rêve... et la science.

Konrad Zuse travaille sur ce qui pourrait être le premier ordinateur moderne : relais mécaniques et électriques, architecture que nous dirions aujourd'hui "classique" ; il invente, fabrique et présente trois générations de sa machine, les Z1, Z2 et Z3, dans l'optique de commercialiser un calculateur.

Il conçoit également en parallèle à ses travaux, une notation mathématique proche de l'algèbre, nommée Plankalkül. Il n'aura été implémenté qu'en 1975, mais sa philosophie et ses idées en font un des tous premiers langages de programmation.

Numérisation, diversification, explosion

Le monde est prêt à être numérisé, et les entreprises se battent pour fournir le matériel et les logiciels nécessaires.

Bien que chacun des modèles matériels et des langages fournis pour les faire fonctionner apportent tous leur pierre à l'édifice, on s'épargnera la liste pour retenir encore une fois un principe de l'innovation : une phase d'exploration et de diversification des approches et des produits avant de se consolider par la suite. Soudages, câbles, unités de calcul, entrées et sorties sur cartes perforées, mémoire interne ou non, type et nombre de relais et portes logiques... On commence d'abord par voir les ordinateurs grandir et grossir, puis réduire sous l'effet du passage au transistor et aux circuits imprimés.

L'âge du logiciel

Pendant un temps, matériel et logiciel avancent main dans la main : le matériel est conçu pour répondre aux besoins de calcul exprimés, le logiciel exploite les spécificités du matériel.

Bientôt, il est trop compliqué d'avoir une vision complète sur toute la chaîne, on choisit des experts que l'on spécialise, et on tente de standardiser l'un pour innover dans l'autre : ce serait dommage qu'un programme de recherche donne un logiciel qui nécessite un jeu d'instructions qui ne se fait plus, ou que l'on vende une machine incapable de faire fonctionner les logiciels du moment.

Standard et spécifique, ouvert et propriétaire

Chaque constructeur tente de pratiquer un "vendor lock-in" pour que chaque conquête, chaque ordinateur vendu et installé, se transforme en revenu perpétuel. Par souci d'optimisation de coûts, que ce soit par les constructeurs pour s'économiser de la R&D et ne pas réinventer la roue, ou par le souhait des clients de ne pas refaire son informatique de zéro, on souhaite malgré tout standardiser.

Ce serait magnifique qu'un logiciel conçu pour une machine tourne sur la génération suivante issue du même constructeur. Ce serait tout aussi intéressant d'avoir un seul langage qui soit compatible sur différentes machines. Assurer à ses clients une compatibilité avec le grand succès commercial du moment est forcément un argument de vente intéressant. Le premier outil pour cela sera le système d'exploitation, pour accéder à ce matériel, et le langage de programmation, afin d'exprimer les instructions.

Langages de programmation

Là encore, de nombreux langages informatiques plus ou moins "célèbres" vont chacun introduire un concept, des fonctionnalités, des abstractions, une manière de s'exprimer, ou même simplement influencer les autres parce qu'ils sont des succès en termes de diffusion et de popularité. La clé est comme toujours une adéquation entre les besoins exprimés de l'époque, les moyens à disposition, et le bras de fer commercial que se livrent chacune des sociétés qui promeut sa plateforme.

L'assembleur permet de parler à la machine, le Fortran (FORmula TRANslation) d'exprimer des formules mathématiques, le Simula, conçu pour faire des simulations, introduit les concepts Objet, le C est le langage de la plateforme Unix, Smalltalk est tout autant un langage que son environnement de développement, et le JavaScript est partout dans nos navigateurs Internet.

... Est-ce que cela a du sens de les comparer ? De les analyser, de voir les possibilités, de croiser les idées, oui. Mais de croire que l'un peut remplacer l'autre dans son usage principal ?

De quelle génération ?

On retrouvera souvent le terme de "génération" de langage informatique.

C'est une présentation souvent difficile et fallacieuse : on s'attendrait à voir une chronologie ou un héritage d'idées, et il y en a bien entendu, mais la "génération" est une manière de répondre aux besoins. Mais puisqu'il le faut, détaillons : la première génération est quasi mécanique ou matérielle, la seconde génération comprend les langages assembleur pour les instructions de chaque machine, et la troisième est celles des "langages de haut niveau".

Difficile de dire quelle abstraction justifie la qualification de langage de quatrième génération ("un plus haut niveau d'abstraction"), voire de cinquième génération ("résolution de problèmes par les contraintes données plutôt qu'un algorithme"). De même, difficile de dire ce qui sépare un langage de 5e génération d'un logiciel bien conçu et prêt à l'emploi, qui aurait simplement un formalisme orienté texte.

Pire encore : un "langage" textuel servant à représenter de la donnée, avec ou sans forme (XML, HTML...), est-il un langage ? Un outil graphique servant à représenter un programme (Scratch...) est-il un langage ?

De quelles philosophies ?

Il serait plus juste de classer et représenter les langages par leur philosophie : un langage système est proche de la machine, les langages objet et/ou fonctionnels proposent leurs abstractions, les DSL (Domain Specific Languages) sont des moyens d'exprimer un problème ou une solution, d'une manière limitée et cadrée par le concepteur de chacun de ces langages, pour permettre une richesse et une concision dans un domaine métier bien particulier.

Rencontre avec un langage

La "première rencontre" avec un langage informatique est sa syntaxe : la sémantique du langage dit ce que l'on peut exprimer et comment, la "grammaire" du langage permet de déterminer ce qu'est un programme composé d'instructions correctement exprimées, mais

puisqu'il faut bien la taper avec du texte, des espaces et des point-virgules, et puisque l'ordinateur est impardonnable au caractère près, c'est à la syntaxe qu'on s'arrête souvent. Par exemple, il n'est pas rare de critiquer Lisp (en blaguant ou non) par le nombre de parenthèses que l'on voit dans un exemple de code.
Un deuxième point très visible et donc très débattu, est son système de typage : pour exprimer les opérations à faire sur les éléments à disposition, encore faut-il savoir à quoi ils sont censés ressembler. Là encore, on parlera davantage de typage explicite (que l'utilisateur doit renseigner) ou implicite (le compilateur ou interpréteur saura deviner), la partie visible, que de typage fort ou faible (converti automatiquement ou non) ou de typage statique (connu à l'avance) ou dynamique (découvert lors de l'exécution). Enfin, si l'on a du code, on souhaitera le tester. Un troisième point visible sera alors la manière de le faire fonctionner : avec une étape de compilation, ou sans ("langage de script"). Ces points ne sont que très rarement structurants, mais ils sont visibles. C'est pourquoi ils forment le gros des débats lorsqu'on parle d'un langage informatique et qu'on le compare aux autres.

Paradigmes, outillage et communauté

En réalité, ces arguments n'ont pas autant d'importance sur le long terme. Si les artisans ont le bon outil pour la bonne tâche, ils savent bien que "le mauvais ouvrier blâme les outils". Dans un projet concret, en équipe et dans la durée, c'est plutôt la liste des paradigmes et philosophies de chaque langage qui importe : l'usage et le pragmatisme l'emportent sur la théorie spéculative. La position d'une parenthèse ou des mots clés sont affaire de goût personnel. Et parler d'un langage, c'est aussi considérer l'ensemble des gens qui l'utilisent, leurs contextes, leurs habitudes et leurs découvertes. L'ensemble des outils disponibles et plus ou moins utilisés, les bibliothèques et frameworks (open-source ou propriétaires), les environnements d'exécution, les outils de test, de livraison...

Qu'est-ce qui a du sens en ce moment ?

Bien sûr, nous vivons et travaillons dans un monde dont la part virtuelle semble grandissante. Tous ces progrès techniques nous ont donné cette chance de s'affranchir des contraintes du "monde physique", pour des gains exponentiels. Modifier un comportement semble à la portée d'un simple if. Analyser des interactions complexes, valider une hypothèse ? À la portée d'une simulation ! On clone un environnement et on rejoue les interactions.
C'est aussi des réflexes qui n'ont plus cours, et le piège permanent d'oublier que dans un monde numérisé et connecté, notre travail n'est pas virtuel mais a des conséquences bien réelles. On se retrouve ainsi plus souvent limité par ses propres idées et conceptions que par une limite physique, ce qui peut causer de la frustration et donner des leçons d'humilité.
Le prix ? Une complexité et un besoin d'abstraction accrus. À tout moment, on doit se reposer la question : qu'est-ce qui a changé ? Qu'est-ce que j'ai appris ? Est-ce que l'arbitrage que nous avons choisi a encore du sens ?
Et plus ce monde va vite, plus ce besoin de se mettre à jour en permanence se fera sentir.

L'ÂGE DES RÉSEAUX

Après la guerre, l'utilité de l'informatisation n'est plus remise en cause. Banques, assurances, lignes aériennes... et de plus en plus de secteurs vont lancer des projets pour numériser leurs processus d'entreprise.
Au-delà du dogme, une fois les données récupérées, en détail et sur des années, l'enjeu sera ensuite de les agréger, puis de les catégoriser, pour agir dessus vite et bien. Plus tard, les particuliers entreront dans la danse. Le système de la ligne aérienne était informatisé et réservé aux aéroports ? Désormais, le particulier lui-même pourra réserver son billet en ligne, voire passer à un billet dématérialisé et oublier complètement l'étape papier. Le monde est de plus en plus connecté, numérisé, et cette double capacité d'analyse et d'action fait la différence entre les leaders, les outsiders en croissance, et les tenants du titre en déclin.

Un ordinateur au travail

Par sa capacité d'investissement, l'entreprise va amorcer le changement. Tout d'abord par les outils qui n'ont de sens qu'avec des fonctions automatisées : robots industriels, aéronautique, imagerie médicale... qui sont autant de raisons d'investir dans la recherche, le matériel et le logiciel adaptés. Puis par le travail de plateaux entiers d'employés sur ordinateur. Les commandes massives d'ordinateurs, d'outils informatiques, de prestations de développement... sont chères, mais apparaissent rentables au vu du temps gagné par un nombre croissant d'employés, partenaires, clients et consommateurs.

Un ordinateur à la maison

Après un cycle réservé aux gros budgets, qui permettent de lancer le mouvement d'optimisation, réduction de coût des équipements, l'informatique se démocratise. On parle d'un ordinateur dans chaque maison.
Bureautique, jeux, médias, puis l'arrivée d'Internet transforment cette TV interactive en fenêtre sur le monde, puis en outil indispensable. Quelques années durant, les particuliers voient en entreprise ce qu'ils pourraient avoir chez eux.
Puis la tendance s'inverse, et ils exigent de trouver en entreprise la simplicité d'usage et le confort que les logiciels grand public leur fournissent à la maison.

Un ordinateur dans la poche

Encore une fois, l'état de l'art, l'économie et la société permettent l'émergence de nouveaux usages : les ordinateurs sont miniaturisés, les batteries sont moins chères et tiennent des charges plus fortes plus longtemps, les réseaux mobiles sont déployés "partout", une partie grandissante de la population utilise Internet.
Autant les PDA avaient trouvé leur niche, autant les smartphones conquièrent leur public en un temps record. Les usages que l'on avait aperçus avec Internet s'adaptent ; les usages qui nécessitaient d'être toujours connectés, réactifs, disponibles sur demande ne sont plus bloqués par le fait d'avoir uniquement la voix ou le SMS à disposition.

Un ordinateur partout

Outils médicaux, voitures, compteurs électriques, mais aussi jouets, thermostats, traceurs d'activité physique... Plus l'électronique est miniaturisée, peu gourmande et bon marché, plus on pourra "améliorer" les objets de tous les jours avec des fonctions numériques et communicantes, et peut-être aussi des fonctions de blocage, de licences, de service d'agrégation des données payants.

Tout ça pour quoi ?

Cet héritage glorieux des sciences et techniques nous laisse aujourd'hui devant une pléthore d'outils, dont certains se remplacent, se complètent, ou se ressemblent un peu trop et font alors l'objet de débats sans fin : on croit que tel outil est remis en cause par l'autre partie, alors qu'il propose d'améliorer telle autre partie de notre métier.
Nos vies vont changer de plus en plus vite, mais avec cette histoire et ces principes de l'évolution en tête, nous allons pouvoir faire un état de l'art puis une projection dans le futur. À bientôt pour la deuxième partie !

•

Backlight de la 3D à la VR

François Tonic

Au cœur de Paris, un studio mise sur la réalité virtuelle aussi bien dans les jeux que les applications professionnelles. Le studio avait fait une forte sensation en 2015 avec le jeu Birdy King Land. Une immersion dans un univers en VR pour Oculus qui a connu un très gros succès. Aujourd'hui, le studio travaille aussi bien pour le marché français qu'à l'international et notamment en Asie qui connaît une explosion des salles d'arcade nouvelle génération.

Pour les fondateurs, Jonathan TAMENE et Frédéric LECOMPTE , la réalité virtuelle est déjà "largement" disponible auprès du grand public. L'un des derniers entrants est Sony avec le PlayStation VR. Les solutions VR mobiles (casques supportant un smartphone) ne rentrent pas dans cette catégorie. La véritable VR passe par des casques de type Oculus ou Vive. Les professionnels sont demandeurs de ces nouvelles technologies immersives, essentiellement dans les secteurs de l'industrie et de l'immobilier. Le studio teste l'ensemble des solutions du marché. "Le HTC Vive est vraiment bien. L'oculus reste un peu en dessous, notamment niveau tracking. Le cardboard, quant à lui, reste un peu gadget mais offre bien souvent la première immersion en VR. Aujourd'hui, le marché que nous visons se concentre sur les événements, les salles d'arcade, les Escape Room…" Analyse Jonathan TAMENE.

Le marché asiatique explose !

En Asie, la réalité virtuelle se déploie dans des salles d'arcade avec jeux et matériels. "On plonge littéralement dans un univers et on sort de son quotidien. Il y a une vraie demande et pourtant, la séance coûte assez cher entre 5 et 20 $. Récemment, HTC a annoncé l'ouverture de milliers de salles équipées en Asie, les Viveport Arcade. La Chine et Taiwan furent les deux premiers marchés.

Nous n'en sommes qu'au début

"La technologie va s'améliorer, et rapidement sans doute. HoloLens est une technologie dingue et orientée réalité mixte. On mappe le virtuel sur l'environnement réel. Sur l'Oculus, la caméra n'est pas trop utilisée pour le moment. Sur le Vive, on perd parfois le tracking et l'espace d'action est restreint à 25m². C'est là que Hololens fait la différence en triangulant sa position et la notion de profondeur intégrée au système de caméra." analyse Jonathan TAMENE

Comme nous l'a précisé l'équipe, un des avantages du Vive est que son système Lighthouse, pour tracker les objets et les mouvements, est ouvert aux développeurs et est « open source », avec des licences dans certains cas. Cette ouverture peut permettre de nouveaux accessoires et logiciels dédiés !

"Avec cette technologie de tracking, la VR est en train de révolutionner un autre domaine : la motion capture dans le cinéma et les jeux ! C'est la mort annoncée de la Motion Capture telle qu'on la connaît aujourd'hui avec le mélange de la VR, du tracking, de la 3D." évoque Jonathan TAMENE

"La VR isole-t-elle ?"

Un studio qui suit les évolutions du marché

Jonathan TAMENE raconte les débuts du studio. Fin 2007, début 2008, Backlight faisait de la 3D mais pas de jeux. C'était le démarrage. L'activité se concentrait sur les projets digitaux, des animations 3D pour le Web. Mais rapidement, le studio travaille étroitement avec Total Immersion, spécialisé en réalité augmentée. Le but : créer du contenu 3D temps réel pour des expériences en réalité augmentée pour la communication ou des parcs d'attraction comme le Futuroscope (France) et Six Flags (USA). Durant plusieurs années, Backlight crée des animations et de la 3D. Puis, les fondateurs voient arriver les premiers kits Oculus Rift.

"On teste, on conçoit quelques petits projets en R&D pour évaluer les technologies et le potentiel. Rapidement, l'idée de faire de la VR dans le studio naît et c'est ainsi que le jeu Birdy King Land voit le jour. Il s'agit d'un ride avec une ambiance, un environnement complet qui donne envie de s'amuser. Le jeu est proposé gratuitement sur le marketplace Oculus. Et le succès arrive très vite pour atteindre 100 000 téléchargements ! Soit environ 50 % du parc d'Oculus mondial. Rapidement, nous avons des dizaines de vidéos sur Youtube, des millions de vues, des articles et peu à peu, le jeu s'impose comme une référence de la VR.

Grâce à ce succès et à cette expérience, le studio développe des compétences VR très pointues et teste les solutions du marché. Et les

premiers projets commencent à arriver. L'un des premiers fut pour l'Union des Industries des Métiers de la Métallurgie pour découvrir la réalité des métiers de cette industrie mal connue. "Dans la tête des jeunes, l'industrie ce ne sont que les hauts fourneaux. En leur offrant une expérience de réalité virtuelle dédiée, les jeunes, la cible, découvrent à quel point ces métiers sont orientés vers les nouvelles technologies." conclut Jonathan TAMENE

En deux ans les projets s'enchaînent avec un catalogue qui s'étoffe de plus en plus faisant de Backlight la référence VR en France, avec une quinzaine de projets en commande pour différentes marques et différents domaines d'activité. Piaget (joaillerie), Point P (matériaux de fabrication), Groupe Huawei (Téléphonie), MSD (Laboratoire pharmaceutique), Somfy (domotique à domicile), SNIA (Aéroports de Paris), The Camp (architecture) voient des projets qui seront développés en montrant ainsi le champ des possibles en réalité virtuelle.

Les développeurs utilisent beaucoup le moteur Unreal Engine et l'outil Blueprint qui permet de créer "rapidement" l'environnement complet et les interactions. Le code est réalisé en C++. Mais le studio travaille aussi sur Unity.

Une évolution de Birdy King Land est en développement et sera proposée sur Vive et Oculus ; il restera gratuit. Une version 2 du jeu est aussi en préparation mais de l'aveu de Jonathan TAMENE, le projet coûte cher en budget et en temps. Une réflexion est en cours sur le modèle économique, même si le studio veut que les gens puissent découvrir cette future version. •

Des **interactions sociales intuitives et naturelles** sont-elles possibles dans la réalité virtuelle ?

Le bras du robot s'étend pour saisir l'objet au moment où celui-ci est déposé par un individu. Ce mouvement, qui semble totalement naturel, repose toutefois sur la capacité du robot à anticiper l'intention de l'individu à partir de l'observation de ses actions. En l'occurrence, l'objectif de l'individu est bien de prendre un objet pour le transmettre au robot, et cette intention se traduit par une expression motrice particulière. Autrement dit, les mouvements que l'on produit sont différents quand on a l'intention d'agir sur un objet pour le transmettre à quelqu'un plutôt que lorsqu'on a l'intention de le réutiliser soi-même par la suite. Cette intention (sociale ou personnelle) influence les caractéristiques cinématiques des productions motrices et est inconsciemment perçue par l'observateur.

L'objectif du projet développé au sein des laboratoires CNRS CRIStAL et SCALab de Lille était d'identifier par des algorithmes mathématiques les intentions dans les actions pour permettre d'interagir de manière fluide et efficace avec un avatar en réalité virtuelle ou avec un robot, et faciliter ainsi l'acceptabilité de la machine par l'homme. En d'autres termes de rendre plus naturelles les interactions avec des systèmes artificiels de tous types.

Pour comprendre ces interactions, les chercheurs ont utilisé un système de Motion Capture (Mocap-Qualisys) et des capteurs ont été placés sur les articulations d'un individu. Lors des mouvements dirigés vers les objets, le Mocap récupère en temps réel les positions et rotations du bras qui se déplace. Un scénario a été développé où l'individu se trouve face à un agent virtuel dans un contexte de bar. La tâche pour l'individu est de prendre et déplacer un verre soit pour le réutiliser lui-même par la suite (intention individuelle), soit pour être servi par l'avatar qui a endossé les habits de barman pour l'occasion (intention sociale). Une approche novatrice et originale a consisté à considérer les trajectoires de l'individu comme des courbes. Chaque trajectoire est ensuite exprimée comme un élément dans un espace des formes. Un algorithme d'apprentissage statistique permet de discriminer et classifier les mouvements selon l'intention qui y est associée. Il permet ainsi d'interpréter les gestes humains et de produire des interactions naturelles entre humains et avatars en réalité virtuelle.

Les applications attendues de cette collaboration originale entre informatique et sciences cognitives se déclinent dans de nombreux domaines : postes de travail dans l'industrie, formation à distance, assistance aux personnes handicapées, jeux interactifs…

Ce projet a reçu le soutien du CNRS et le Programme d'Investissements d'Avenir (PIA), Agence Nationale pour la Recherche (grant ANR-11-EQPX-0023), European Founds for the Regional Development (Grant FEDER-Presage 41779). Nous tenons à remercier Maxime Devanne docteur en Informatique, François Quesque docteur en psychologie et Anis Kacem doctorant en informatique à l'Université de Lille, ainsi que Davide Guerrero et Ylenia Esposito étudiants en master de psychologie à l'Université de Naples pour leurs contributions à la réalisation de ce projet.

Actus VR

QUI VEUT D'UN STANDARD ?

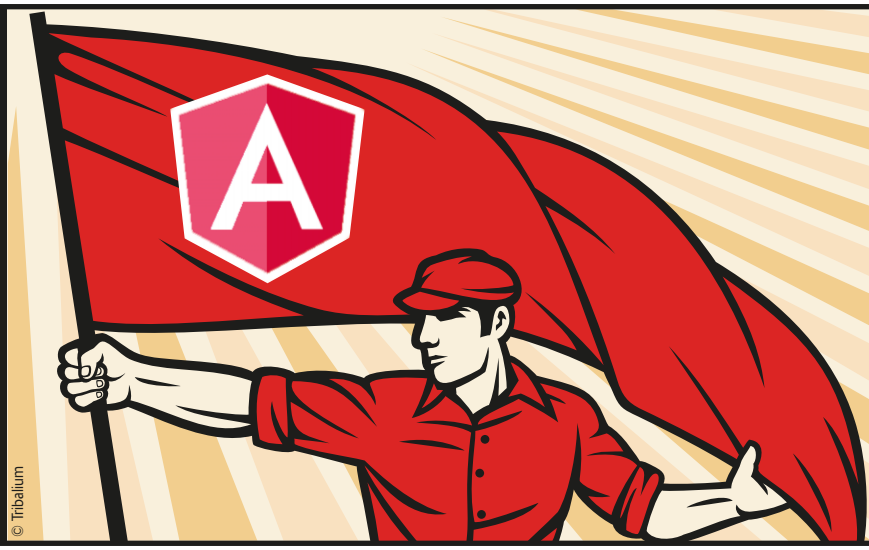
Le Khronos Group souhaite établir des standards pour la réalité virtuelle et donc harmoniser les fondations techniques. Aujourd'hui, chaque constructeur fait un peu ce qu'il veut, même si des plateformes comme Unreal et Unity unifient à la manière le développeur. Mais les casques utilisent des SDK, API, systèmes différents. Le marché n'en est qu'à ses débuts, et, comme toujours, les acteurs créent leurs propres initiatives. Google propose son propre écosystème, Microsoft veut des casques plus abordables basés sur ces technologies et outils, etc. Si tout le monde ou presque se dit heureux de cette annonce de Khronos le chemin sera long.

EYDOLON : RÉSEAU D'ESPACES DE LOISIRS VR

Un groupement d'entreprises travaillant sur la VR, le VR Connection, a ouvert un réseau d'espaces de loisirs et d'expériences autour de la VR : Eydolon. L'objectif est d'ouvrir une vingtaine d'espaces VR pour le grand public en France. Le premier espace a ouvert mi-décembre à Lyon. Trois zones sont proposées :

- L'Arena : autour des technologies Vive et les interactions possibles, et l'expérience, notamment en mode multi-utilisateur
- Immersion : on parcourt un temple ancien.
- Flying experience (début 2017) : simuler un vol libre.

LA NOUVELLE RÉVOLUTION DU WEB



© Tribalium

Parlons peu, mais parlons concrètement. Arrêtons de parler de web 2, 3 ou 4, le développeur Web cherche du concret. Plusieurs évolutions sont importantes et promettent des applications Web de nouvelles générations ou du moins avec des fondations bien plus pertinentes qu'hier.

Pour ce dossier spécial, nous avons sélectionné plusieurs technologies très intéressantes et que vous devez absolument commencer à regarder, si ce n'est déjà fait :

- ***Les progressive web apps (PWA)***
- ***WebGL***
- ***Angular (ex-2)***
- ***Le framework Laravel***

Vous allez me dire que ce n'est pas forcément très révolutionnaire. Et pourtant, PWA marque une évolution réelle dans les apps Web mobiles. WebGL démontre une fois de plus que le navigateur joue un rôle central dans de nombreuses évolutions technologiques. Angular, ex-Angular 2, a perdu son JavaScript, mais a gagné une refonte en profondeur. Oui la rupture avec AngularJS est là, mais n'était-ce pas le prix à payer pour continuer à évoluer ?

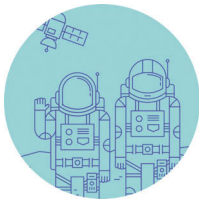
Enfin, PHP démontre avec Laravel qu'il peut toujours évoluer et aider le développeur. Ce framework fait beaucoup parler de lui en ce moment, il était temps de s'y intéresser.

La rédaction.

Les **Progressive** Web Apps

Partie 1

La seconde chance du Web dans l'univers du mobile



SIMON GOMBAUD
Directeur artistique
Made on Mars

Au milieu de l'année 2007, voilà déjà presque 10 ans, une révolution se met en place avec l'arrivée du tout premier iPhone. Les utilisateurs ont un nouveau terminal avec lequel ils peuvent naviguer sur Internet. À l'époque, le Web n'est pas préparé à cette révolution. Les sites ont encore une majorité d'éléments en Flash qui ne sont pas supportés par l'iPhone. Steve Jobs annoncera d'ailleurs 3 ans plus tard dans sa célèbre lettre ouverte "Thoughts on Flash" qu'il n'a aucunement l'intention de changer ça. Été 2008, Apple dévoile l'iPhone 3G et l'App Store.

Les gens peuvent désormais trouver un choix immense d'applications leur permettant de jouer, lire les actualités, trouver une recette de cuisine… Comme le répète la publicité de l'époque, il y a "une application pour à peu près tout". En face, le Web n'a toujours pas trouvé de solution pour répondre au besoin d'un nouveau mode de navigation sur Internet. Il faudra encore plusieurs années pour que des solutions propres viennent se mettre en place avec le HTML5, le CSS3 et des librairies JavaScript orientées mobiles. Puis, ces dernières années, des solutions comme Apache Cordova et Adobe PhoneGap, sont apparues pour générer des applications mobiles qu'il est possible de publier sur les stores bien qu'elles soient développées en technologies Web. Finalement, il aura fallu attendre que les téléphones aient une puissance acceptable pour faire tourner le web serveur et charger le JavaScript qui n'offrait pas une expérience proche du natif par le passé.

C'est ainsi que lors d'un dîner avec sa femme Frances Berriman, l'ingénieur Google Alex Russell marque sur un bout de papier les éléments clés d'une expérience mobile et les prérequis qu'une Web App se doit d'implémenter pour être dite mobile-friendly. L'article original est visible à cette url (en anglais): https://infrequently.org/2015/06/progressive-apps-escaping-tabs-without-losing-our-soul/

Ce manifeste amènera une nouvelle vision du Web sur mobile, Alex et sa femme appelleront cette approche les "Progressive Web Apps".

Ci-dessous la liste traduite en français :

- **Responsive** : permettre une utilisation sur tous les types d'appareil.
- **Interactions aux allures natives** : avoir une cohérence de structure rappelant le principe des applications natives ainsi que leurs interactivités.
- **Indépendant du réseau** : utiliser des Services Workers pour permettre à l'application de fonctionner hors ligne.
- **A jour** : avoir une mise à jour des données de manière transparente pour l'utilisateur.
- **Sécurisée** : être servie via le protocole TLS.
- **Peut être installée** : proposer à l'utilisateur d'installer l'application sur sa page d'accueil.
- **Peut être partagée** : permettre le partage de l'application simplement par un lien URL.
- **Référencée** : être référencée comme application sur un moteur de recherche.
- **Favorise le réengagement** : utiliser les fonctionnalités de notifications.

Tout au long de l'article, nous détaillerons ces différents points qui font appel, pour la plupart, à des concepts très récents des technologies Web. Nous accompagnerons ces explications par la construction d'une Progressive Web App très basique. L'idée de l'application est de créer un compteur collaboratif. Les gens ajoutent des cœurs à un compteur en cliquant sur un bouton. L'application doit être utilisable de manière hors ligne et une fois la connexion retrouvée, venir mettre à jour les cœurs ajoutés hors ligne. Nous avons créé un reposit sur Github qui permet de se lancer rapidement dans des petits projets comme celui-ci. Ce boilerplate d'Express (http://expressjs.org) utilise Gulp pour écouter les fichiers js, css et html afin de recharger automatiquement la page à chaque modification. Pour l'installer :

```
git clone https://github.com/MadeOnMars/Express-boilerplate.git myFirstPWA
cd myFirstPWA
npm install
npm install --global gulp
gulp
# Vous pouvez voir le projet à cette URL : http://localhost:3000
```

Responsive

Le Responsive Web Design (RWD) est la première notion pour appréhender une expérience Web mobile réussie. Son objectif est de rendre le site adaptable à toutes les tailles et résolutions d'écrans, avec une expérience de lecture et de navigation optimale. Ethan Marcotte a été le premier à introduire le terme de « Responsive Web Design » dans un article de "A List Apart" en mai 2010 (http://alistapart.com/article/responsive-web-design).

Concrètement, le Responsive Design consiste à enrichir la feuille de style CSS du site ou la Web app avec des Medias Queries. En CSS2, l'attribut media permettait déjà de spécifier un media de destination pour chaque feuille de style. Ainsi, il est possible d'appliquer une mise en page spécifique pour différents contextes : principalement l'écran et l'impression. Depuis CSS3, ces directives peuvent être directement intégrées à la feuille de style, ce sont les Medias Queries. Grâce à la règle **@media**, il est possible de définir le media de destination (screen, print, tv…), mais également d'autres critères comme la hauteur et la largeur de la zone d'affichage du périphérique, son orientation…

Mise en pratique

Nous allons commencer par mettre en place la structure html de notre app dans le fichier public/index.html :

```
<header>
 <nav id="menu">
  <ul>
   <li> ... </li>
   <li> ... </li>
   <li> ... </li>
   <li id="hamburger"> ... </li>
  </ul>
 </nav>
 <ul id="param"> <button id="subscribeBtn" disabled>Recevoir PNs</button> </ul>
</header>

<div id="mobileNavigation">
 <div class="inner">
  <div id="close"> Close </div>
  <div id="illu"> ... </div>
  <ul id="mobileNav"> ... </ul>
  <ul id="mobileParam"> ... </ul>
  <div id="mobileFooter"> ... </div>
 </div>
</div>

<div id="counter">
 <span id="count">0</span>
</div>

<section id="mainContainer">
 <div id="content">
  <div id="legend"> ... </div>
  <div id="btn"> <button>+1</button> </div>
 </div>
</section>

<footer> ... </footer>
```

Pour des questions de lisibilité, le code a été simplifié. Vous le trouverez dans son intégralité sur GitHub : https://github.com/MadeOnMars/lovelovelove

Nous allons ensuite définir les styles de l'arche du site (les éléments de l'interface qui seront présents sur toutes les pages), pour les plus grands écrans. Dans le fichier public/css/style.css :

```
html, body {
 margin:0;
 padding:0;
}

body {
 font-family:'Lato', sans-serif;
```

```
  font-weight:100;
  background:#3b395e;
  color:#ffffff;
}

header {
  background:#2e2e49;
  padding:20px;
  position:relative;
  z-index:30;
}

header #menu {
  float:left;
}

header #menu ul li#hamburger {
  margin:0;
  display:none; /* On cache le bouton burger sur desktop */
}

header #param {
  text-align:right;
}

header #param li span {
  display:inline-block;
  vertical-align:middle;
  margin-left:10px;
}

#mobileNavigation {
  height:100%;
  width:90%;
  display:table; /* Centrage vertical */
  table-layout:fixed;
  position:fixed;
  z-index:100;
  top:0;
  left:0;
  background:#2e2e49;
  transform: translate3d(-100%, 0, 0); /* On fait sortir le menu du viewport */
  transition: all 0.3s ease-in;
}

#mobileNavigation .inner {
  position: relative;
  display: table-cell; /* Centrage vertical */
  vertical-align: middle;
}

body.navigation #mobileNavigation {
  transform: translate3d(0, 0, 0); /* On réinitialise la position du menu */
}

footer {
  padding:20px;
  text-align:center;
  position:fixed;
  right:0;
  bottom:0;
  font-weight:600;
  font-size:14px;
  color:#25253a;
  z-index:15;
}

footer a {
  text-decoration:none;
  color:#25253a;
}
```

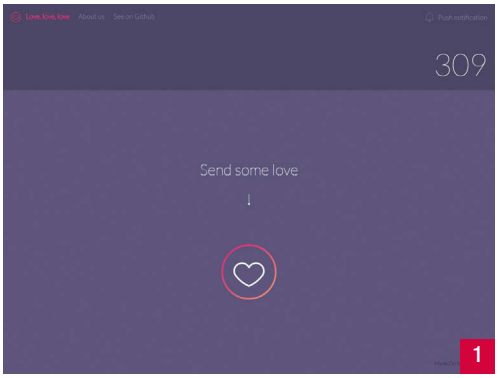
On définit ensuite les styles du contenu de l'app :

```
#counter {
  background:#2e2e49;
  padding:80px 20px 20px 20px;
  text-align:right;
  position:relative;
  z-index:20;
}

#counter span {
  font-size:100px;
  display:block;
}

#mainContainer {
  padding:0;
  height:calc(100% - 288px);
  width:100%;
  overflow:hidden;
  position:absolute;
  display:table;
  table-layout:fixed;
  top:288px;
  left:0;
  z-index:10;
}

#mainContainer #content {
  display:table-cell;
  vertical-align:middle;
  text-align:center;
  height:100%;
  width:100%;
  padding:40px 0;
}

#mainContainer #content #legend h1 {
  font-weight:100;
  font-size:50px;
  text-align:center;
  margin:0 0 40px 0;
```

```
}

#mainContainer #content #btn {
  margin-top:90px;
}

#mainContainer #content #btn svg {
  width:140px;
  padding:10px;
  cursor:pointer;
}
```

Nous obtenons ce résultat sur un écran de 1280px de large : [Fig.1]. A partir de 768px, l'écran devient trop petit pour afficher le menu dans son intégralité. On passe donc en navigation mobile :

```
@media screen and (max-width: 768px) {
  /* On cache le menu et on affiche le bouton Burger */
  header #menu ul li,
  header #param li span,
  footer {
    display:none;
  }
  header #menu ul li#hamburger {
    display:block
  }
  /* On adapte les tailles de texte et les marges */
  header {
    padding:20px 20px 0 20px;
  }
  #mainContainer #content #legend h1 {
    font-size:24px;
    font-weight:300;
  }
  #counter span {
    font-size:60px;
  }
  #counter {
    padding:20px;
  }
  #mainContainer {
    height:calc(100% - 160px);
    top:160px;
  }
}
```



1

Cela nous permet d'adapter l'affichage sur tablette : [Fig.2]. Et enfin, nous adaptons quelques styles pour prendre en charge les mobiles à partir de 480px :

```
@media screen and (max-width: 480px) {
  #mainContainer #content #btn {
    margin-top:30px;
  }
  #mainContainer #content #legend h1 {
    margin:0 0 20px 0;
  }
}
```

[Fig.3]

Au service de l'expérience utilisateur

Le Responsive Design est en place depuis quelques années maintenant, il est devenu indispensable dans tout projet Web, et en particulier après le lancement de l'algorithme Pigeon de Google (juillet 2014 aux États-Unis, en juin 2015 en France). Désormais, Google favorise dans ses résultats de recherche les sites adaptés au mobile : [Fig.4]

4

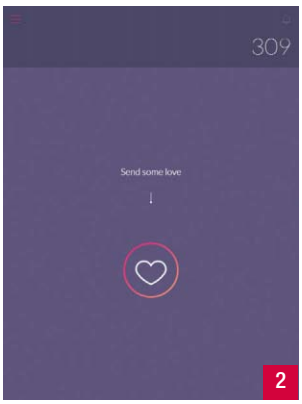
Avec son algorithme, Google a accéléré le processus de transition vers un Web adaptatif, une bonne chose donc pour ses utilisateurs. De même, il y a fort à parier que les moteurs de recherche réserveront demain un traitement de faveur pour les Progressive Web Apps. En effet, au-delà du fait qu'une PWA est forcément responsive, elle rassemble également plusieurs bonnes pratiques à adopter pour proposer la meilleure expérience possible sur mobile.

Interactions aux allures natives

L'objectif d'une PWA est de se rapprocher au maximum d'une expérience d'application native. Avec l'apparition des smartphones, de nouveaux standards se sont imposés en matière de navigation et de structuration de l'information. Il a fallu repenser le design d'interface sur des écrans plus petits. Il a fallu également apprendre à exploiter les possibilités que le tactile a apportées.

Le design mobile

L'exemple le plus parlant est sans doute le « Burger Menu » (« Hamburger Menu » ou « Split View »). Il y a quelques années, cette icône constituée de 3 traits parallèles ne vous évoquait pas grand-chose. Mais elle est apparue petit à petit dans les interfaces mobiles, jusqu'à s'imposer comme le symbole du menu de navigation. Son but premier est de cacher le menu de navigation sur mobile, pour éviter d'ajouter de la hauteur à une page. Généralement placé en haut à gauche de l'écran, il permet de déployer le menu au clic. Pour la petite histoire, la première icône "Burger Menu" a été conçue en 1981 par Norm Cox pour sa société Xerox Star. Il explique que l'icône devait



2

3

être extrêmement simple et mémorisable, et traduire l'idée d'un menu sous forme de liste. Le Burger Menu est devenu un élément système sur les OS mobiles. Mais on voit qu'il est désormais présent (avec quelques variantes) dans l'interface des OS desktop, des navigateurs Web, logiciels… C'est un signe que l'icône est maintenant considérée comme universellement comprise, mais il faut cependant savoir l'utiliser. En 2015, la compagnie Moovweb a analysé 50 sites mobiles afin de mettre en évidence l'engagement des utilisateurs auprès du burger menu. Il apparaît que seuls 20% des utilisateurs cliquent sur le bouton burger, contre 30% sur un lien de menu apparent. L'étude montre aussi qu'un bouton burger est beaucoup plus cliqué s'il dispose d'un label "Menu", qui identifie directement l'action associée. [Fig.5 et 6]. Malgré ces réserves, il apparaît que le Burger Menu reste la solution la plus élégante pour afficher un menu constitué de nombreux items, sur une interface mobile. Si le menu est plus réduit, d'autres solutions existent comme les Tabs par exemple. [Fig.7]

Avec une PWA, on cherche à reproduire ces éléments système pour garantir une expérience proche du natif. Concernant le design, deux approches sont possibles. Soit le designer décide de conserver une identité propre jusque dans les détails de l'interface (icônes, marges, tailles de texte), soit il fait le choix de suivre strictement les guidelines des différents OS mobiles. L'intérêt de suivre les guidelines est de proposer à l'utilisateur une expérience de navigation qui lui est déjà familière.

C'est le cas par exemple avec le Material Design de Google. Voilà ce qu'on peut lire sur le site officiel (https://material.google.com/) : "Nous avons établi un langage visuel pour nos utilisateurs qui exprime la simplicité, l'innovation avec une inspiration scientifique et technologique. Nous l'avons appelé Material Design". Le Material Design est donc un langage visuel à part entière, étudié pour garantir ergonomie, confort de lecture et de navigation. Toutes les apps que sort Google sont conçues selon ces fameuses guidelines. Les utilisateurs d'Android ont donc l'habitude de cette ergonomie. En mettant à la disposition du public ces règles de design, Google participe à l'unicité des apps tournant sur son système. Pour les développeurs sans compétences graphiques, utiliser un template mobile proposé par Google (Material Design lite : https://getmdl.io/) est une aubaine. Cela permet de garantir un design clair et éprouvé à moindre coût. [Fig.8]. Voici quelques exemples de components proposés dans le kit Material Design lite : [Fig.9]

Les interactions

Reproduire une expérience d'application native ne se cantonne évidemment pas qu'au design. L'expérience utilisateur sur mobile est également définie par les interactions rendues possibles grâce à l'accéléromètre des téléphones, aux notifications, et bien sûr à l'écran tactile.

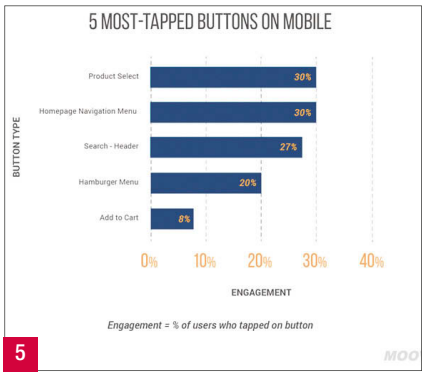
Avec une PWA, on va devoir s'appuyer sur JavaScript pour détecter une gestuelle. Imaginons que je souhaite détecter le slide de gauche à droite pour déployer le burger menu (gestuelle système sous Android). Nous allons utiliser une librairie JavaScript pour détecter l'évènement "swipe" : http://hammerjs.github.io par exemple. Ce plugin permet de détecter plusieurs gestuelles comme le "swipe", le "pinch/zoom", ou encore de mesurer le temps de pression du doigt pour différencier un "tap" d'un "press".

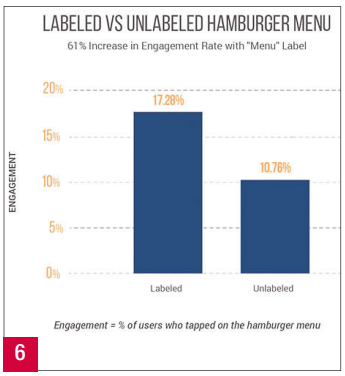
Mise en pratique

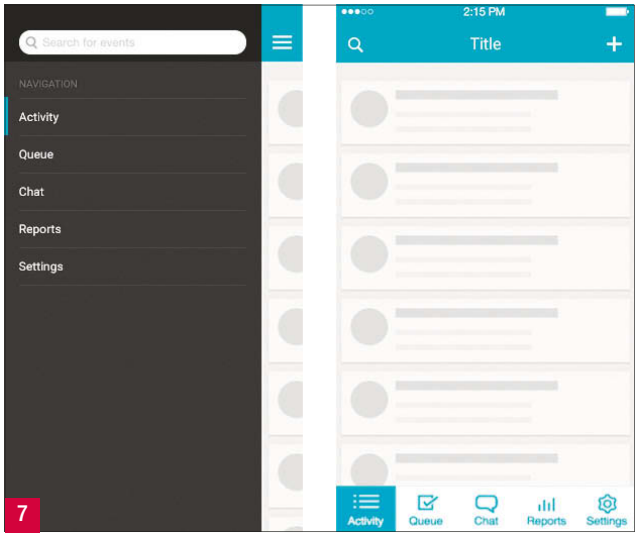
Revenons à notre exemple d'application, commençons tout d'abord par ajouter les interactions de base pour faire fonctionner le compteur. Ajoutons le code ci-dessous dans le fichier public/js/app.js

```
var hamburgerBtn = document.getElementById('hamburger');
var closeBtn = document.getElementById('close');
var loveBtn = document.getElementById('btn');
var countElement = document.getElementById('count');

// Par défaut le compteur est à zéro
var count = 0;

// Lors d'un clic sur l'élément burger on ajoute une classe 'navigation'
// au body qui permet de déployer le menu
hamburgerBtn.addEventListener('click', function() {
 document.body.classList.add('navigation');
}, false);

// Lors d'un clic sur l'élément close on supprime la classe 'navigation'
// au body qui permet de fermer le menu
closeBtn.addEventListener('click', function() {
 document.body.classList.remove('navigation');
}, false);

// Au clic sur le bouton d'incrément on ajoute un coeur au compteur
// puis on met à jour le span "count"
loveBtn.addEventListener('click', function() {
 count++;
 countElement.innerText = count;
}, false);
```

Ce code est très simple, on ajoute un écouteur de clic sur le bouton prin-







cipal. Lors d'un clic, on modifie le texte de la span "count" par la valeur du compteur. Le déploiement du menu est déclenché par la class "navigation" sur le body. Nous ajoutons pour l'instant la class au clic sur le bouton "hamburger". Maintenant, pour ajouter la prise en charge des gestuelles, nous ajoutons la librairie hammerjs dans notre dossier js et nous appelons le script dans index.html

```
...
</body>
<link rel="stylesheet" href="/css/style.css">
<script src="/js/hammer.min.js"></script>
<script src="/js/app.js"></script>
...
```

Puis dans le fichier public/js/app.js, on créé une instance de la librairie et on lui fait écouter l'élément body

```
var hammertime = new Hammer(document.body);
```

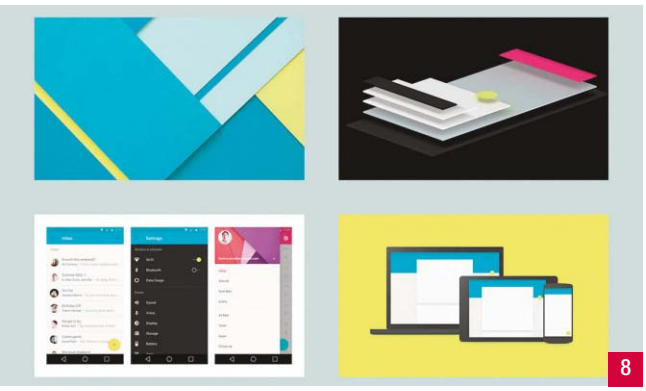
Plus bas, nous ajoutons ces deux sections de code qui se déclencheront lors d'un swipe gauche ou droite

```
hammertime.on('swipeleft', function() {
   document.body.classList.remove('navigation');
});
hammertime.on('swiperight', function() {
  document.body.classList.add('navigation');
});
```

Pour aller plus loin et se rapprocher davantage des comportements système, il existe une librairie très puissante appelée Ionic (http://ionicframework.com/). L'intérêt de ce framework est qu'il intègre tout ce que nous avons vu jusqu'ici : des composants de design (templates, boutons, menus, icônes…), la détection de gestuelles et des performances optimisées. Au niveau gestuelle, le framework a la qualité de reproduire fidèlement un comportement natif. Bluffant pour une techno Web !
Pour gérer les parties événement, routing et data, des frameworks comme React ou Angular permettront d'avoir une gestion du code et des performances excellentes.

Les performances

En comparaison avec un site Web, une application native sera toujours plus fluide et réactive. Et c'est là tout l'enjeu des PWA : le gain en performance. Une Progressive Web App doit être instantanée, sans temps de chargement, pour égaler l'expérience native. Le bon réflexe est d'adopter une architecture Shell + Content (ou coquille + contenu). On va chercher à traiter séparément l'App Shell, c'est-à-dire l'infrastructure / l'interface de l'app, et le contenu de l'app. Les fichiers html, CSS et JavaScript minimums nécessaires à l'affichage de l'interface seront chargés à la première visite, puis passés en cache. La PWA n'a plus qu'à charger le contenu à chaque visite, avec selon la complexité de l'app, ses styles CSS et scripts JS associés. Cela représente un gain énorme en rapidité.


8

Indépendant du réseau

Le point fort indéniable des applications natives de l'époque face au Web était leur utilisation hors connexion Internet. Ceci est rendu possible aujourd'hui grâce aux service workers.
Le service worker est un script Javascript qui va être exécuté en arrière-plan lorsqu'un utilisateur va visiter votre PWA. Ses particularités sont les suivantes :

- Il ne peut pas accéder au DOM mais peut communiquer avec les pages via la méthode postMessage ;
- Il peut intercepter les requêtes réseaux et ainsi modifier le comportement de ces dernières ;
- Il marche uniquement sur des sites Web en HTTPS ou localhost comme nous le verrons plus bas (dans la section **Sécurisée**).

Notez que pour le moment les services workers sont supportés par Chrome, Firefox et Opéra. Edge est en cours de développement et Safari commence seulement à envisager son intégration.

Mise en pratique

Reprenons notre exemple du compteur. Nous allons utiliser un service worker pour mettre en cache certains fichiers pour que notre application fonctionne sans connexion. Tout d'abord, nous allons ajouter un fichier JavaScript qui va correspondre au code de notre service worker. Nous pouvons l'appeler sw.js et y ajouter le code de test ci-dessous :

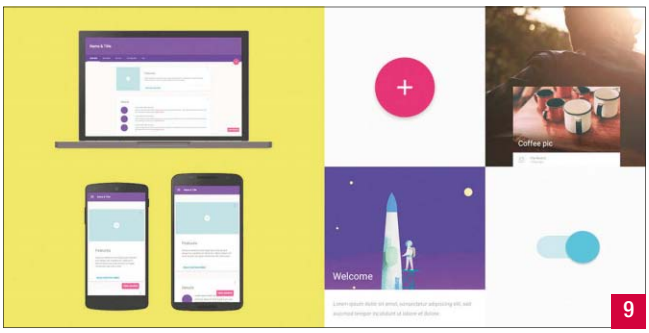
```
console.log('Je suis le service worker');
```

Puis dans le fichier public/js/app.js nous ajoutons le code ci-dessous permettant d'enregistrer notre service worker. Dans le haut du fichier on ajoute une variable globale reg qui va conserver l'enregistrement :

```
var reg;
```

Puis tout en bas de notre fichier public/js/app.js :

```
// On regarde si le service worker est supporté par le navigateur
if ('serviceWorker' in navigator) {
  // On enregistre notre script sw.js
  navigator.serviceWorker.register('sw.js').then(function() {
    return navigator.serviceWorker.ready;
```



9

```
  }).then(function(serviceWorkerRegistration) {
    reg = serviceWorkerRegistration;
    console.log('Enregistrement du SW avec succès.');
  }).catch(function(error) {
    console.log('Une erreur est survenue.', error);
  });
}
```

Ce code commence tout d'abord par vérifier si le navigateur que nous utilisons propose bien le service worker. Si oui, nous tentons d'enregistrer notre script sw.js. Nous pouvons vérifier le résultat de notre console pour voir que tout s'est bien passé. [Fig.10]
Pour information, l'onglet **Application** du developer tool de Google Chrome vous donne aussi des informations et options très pratiques relatives au service worker (SW). [Fig.11]
Notre service worker ne sert pas à grand-chose pour le moment. Ajoutons du code dans notre fichier sw.js afin de lui dire de mettre en cache certaines ressources.

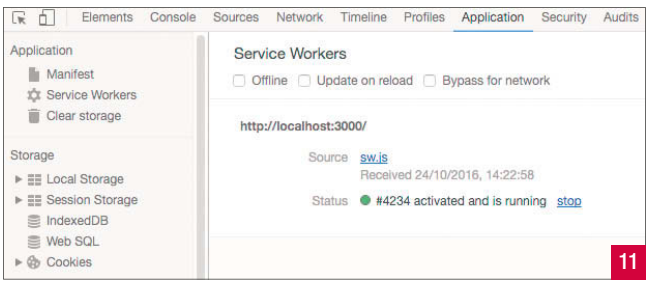
```
var CACHE_NAME = 'my-site-cache-v1';
var urlsToCache = [
  '/',
  '/css/first.css',
  '/css/style.css',
  '/js/hammer.min.js',
  '/js/app.js'
];

self.addEventListener('install', function(event) {
  event.waitUntil(
    caches.open(CACHE_NAME)
      .then(function(cache) {
        console.log('Cache prêt');
        return cache.addAll(urlsToCache);
      })
  );
});
```

Nous ajoutons un écouteur lors de l'événement d'installation du service worker. Ainsi lors de son déclenchement, nous ajoutons dans un cache appelé **my-site-cache-v1** la liste des fichiers contenus dans le tableau **urlsToCache**.
Lancez votre projet, vous verrez que le cache a bien été exploité. Par contre, si on coupe la connexion, l'application ne s'affichera pas. En effet, pour le moment nous avons seulement dit au service worker de mettre dans sa mémoire certains fichiers. Maintenant nous devons lui dire de les récupérer. Pour cela nous allons ajouter un écouteur sur l'événement **fetch**. Le service worker va fonctionner comme un proxy, il va venir écouter chaque requête et nous aurons donc la main sur la réponse. Vous comprenez la puissance des services workers ? D'où l'importance de sécuriser son application et donc la nécessité de servir l'application en https. Si l'une des requêtes correspond à un fichier qui a précédemment été mis dans son cache, on le retourne. Sinon on laisse la requête chercher le fichier sur le réseau. Ainsi avec ce système, si nous n'avons plus de connexion à Internet, l'application s'affichera quand même. Et de même avec une connexion, le service worker va privilégier le cache, ce qui permettra une accélération de l'affichage de l'application.

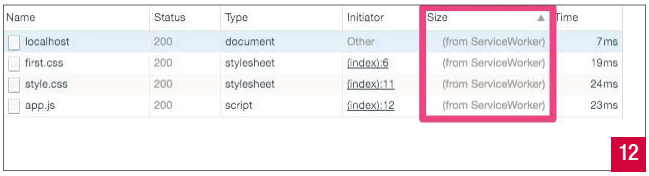
10

```
self.addEventListener('fetch', function(event) {
  event.respondWith(
    // On cherche si la requête correspond à un élement mis en cache
    caches.match(event.request)
      .then(function(response) {
        if (response) {
          // La requête correspond à un fichier en cache, on le retourne.
          return response;
        }
        // Sinon on renvoie la requête qui tentera de le trouver sur le réseau
        return fetch(event.request);
      })
    );
});
```

Vous pouvez maintenant recharger la page et vérifier que votre SW a pris en compte votre nouveau fichier sw.js.
Maintenant pour simuler une coupure d'Internet, stoppez votre web serveur (Ctrl+c si vous utilisez gulp). Rechargez la page, elle doit s'afficher parfaitement.
Vous pouvez observer dans les analyses réseaux que les fichiers en question ont été servis par le SW. De même si vous relancez le serveur et rechargez la page, vous verrez que ces mêmes fichiers sont servis par le réseau. [Fig.12]
Petite remarque si vous voulez mettre à jour vos ressources. En effet, il faut trouver un moyen de modifier les ressources enregistrées en local chez l'utilisateur. Différentes approches existent pour pouvoir mettre à jour un fichier en particulier, mais pour faire au plus simple la méthode de base est de changer le nom du cache c'est-à-dire **var CACHE_NAME = 'my-site-cache-v1';** en augmentant par exemple le numéro de version **var CACHE_NAME = 'my-site-cache-v2';**
A chaque nouvelle connexion, si le SW remarque une modification du fichier sw.js, il va automatiquement repasser en phase d'installation et donc recréer un nouveau cache. Néanmoins, le nouveau service worker ne prendra le contrôle qu'une fois que l'ancien service worker aura été stoppé. C'est-à-dire, une fois que la page aura été fermée.
Comme le cache a une taille limitée mais pas encore définie précisément



11



12

par les différents navigateurs, il est bien de supprimer tous les caches non utilisés. Ainsi une fois qu'un nouveau cache a été activé, l'évènement "activate" est propagé. Il nous suffit donc d'ajouter un écouteur dans notre fichier sw.js et de supprimer les différents caches sauf celui que nous avons défini précédemment. Ici le tableau cacheWhiteList contiendra la liste des caches à conserver.

```
self.addEventListener('activate', function(event) {

 var cacheWhitelist = [CACHE_NAME];

 event.waitUntil(
  caches.keys().then(function(cacheNames) {
   return Promise.all(
    cacheNames.map(function(cacheName) {
     // On supprime tous les caches sauf ceux inscrits dans le tableau cacheWhitelist
     if (cacheWhitelist.indexOf(cacheName) === -1) {
      return caches.delete(cacheName);
     }
    })
   );
  })
 );
});
```

Voilà, avec ces quelques lignes de code, nous avons déjà une application qui peut être utilisée sans connexion Internet. Par contre, vous avez sûrement remarqué que le compteur se remet toujours à zéro. Nous allons voir dans le prochain chapitre comment gérer la persistance des données.

A jour

Nous avons vu comment permettre à la structure de l'application d'être accessible en mode hors ligne. Maintenant il serait intéressant d'être capable de récupérer les données et de pouvoir les exploiter même sans Internet. Puis, une fois la connexion retrouvée, qu'elles puissent se synchroniser. Il faudra tout d'abord stocker les données dans une base côté serveur mais il faudra aussi les stocker côté client. Lors du mode hors ligne, cette base locale viendra prendre le relais pour enregistrer les données. Tout d'abord, on va commencer par permettre à l'application d'enregistrer l'avancement du compteur en mode en ligne. Pour ça, on va modifier notre fichier app.js (serveur web en Node.js) pour qu'il manipule un fichier count.json qui aura comme contenu **{count:0}**. La librairie **fs** de Node.js va nous permettre d'écrire sur le disque. Cette solution n'est pas adaptée à une utilisation en production mais permet de simplifier notre exemple. L'API permettra de récupérer la valeur initiale et d'incrémenter la valeur du compteur en base de données.
Toujours par souci de simplicité, nous allons créer une seule méthode d'incrément à laquelle nous passerons une valeur numérique correspondant à l'incrément nécessaire du compteur. Un appel à l'adresse **/add/1** permettra d'incrémenter d'une unité et un appel à **/add/0** nous retournera la valeur du compteur sans modification. Ajoutons le fichier count.json à la racine du projet avec le contenu ci-dessous :

```
{
   "count": 0
}
```

Dans le fichier app.js on ajoute un require et une route.

```
var fs = require('fs');

app.get('/add/:id', function(req, res){
 // On stocke le paramètre de requête dans une variable
 var increment = parseInt(req.params.id) || 0;
 // Si ce n'est pas un nombre ou que ce dernier est inférieur à zéro
 // on retourne une erreur
 if(Number.isNaN(increment) || increment < 0){
  res.status(500).json({status: 'err'});
  return;
 }
 // Tout semble bon, on ouvre le fichier qui tient le compteur
 fs.readFile('./count.json', 'utf8', function(err, data){
  if (err) {
   res.status(500).json({status: 'err'});
   return;
  }

  // On récupère la valeur présente
  var counter = JSON.parse(data);

  // Pour économiser des écritures si l'incrément est nul on retourne la
  // réponse
  if(increment === 0){
   res.json({status: 'ok', counter});
   return;
  }

  // On incrémente de la valeur reçue et on écrit dans le fichier
  counter.count += increment;
  fs.writeFile('./count.json', JSON.stringify(counter, null, 4), function(err, data){
   if (err) {
    res.status(500).json({status: 'err'});
    return;
   }
   res.json({status: 'ok', counter});
  });

 });
});
```

Maintenant modifions notre fichier public/js/app.js pour venir appeler notre API et mettre à jour la base de données lors d'un clic. Au chargement de la page nous appellerons aussi l'API en passant la valeur 0 afin de récupérer la valeur initiale. Voici le code complet de public/js/app.js

```
var xhr = new XMLHttpRequest();

// Cette fonction permet d'appeler les différents points d'accès de notre api
// tout en passant un callback pour modifier les UIs une fois le résultat reçu
function api(action, value, cb){
 var path = '/';
 switch (action) {
  case 'add':
   path += 'add/'+value;
```

```
    break;
   default:
    path += 'add/0';
  }
  xhr.onreadystatechange = function () {
   if (xhr.readyState === 4 && xhr.status === 200) {
    var response = JSON.parse(xhr.responseText);
    if(response.status === 'ok'){
     cb(null, response);
    } else {
     cb('Something is broken.', null);
    }
   }
  };
  xhr.open('GET', path);
  xhr.send();
}

// On demande à l'API la valeur initiale
api('add', 0, function(err, res){
 if(err){return;}
 counter = countElement.innerText = res.counter.count;
});
```

Nous modifions aussi l'écouteur du clic sur le bouton incrément comme ci-dessous :

```
loveBtn.addEventListener('click', function() {
 // On appelle l'api pour incrémenter le compteur côté serveur.
 // On met à jour notre UI par rapport à la réponse du serveur.
 api('add', 1, function(err, res){
  if(err){return;}
  counter = countElement.innerText = res.counter.count;
 });
}, false);
```

Avec ce code, notre application est entièrement fonctionnelle en mode en ligne et pour tout navigateur. Le principe de "progressive enhancement" est respecté puisque nous créons l'application afin qu'elle soit fonctionnelle sur tous les navigateurs même les plus anciens. Puis, nous lui ajoutons des fonctionnalités plus récentes pour permettre aux personnes avec des navigateurs plus récents d'avoir une meilleure expérience. C'est d'ailleurs de ce terme **progressive enhancement** que vient le terme progressive des PWAs. Maintenant, nous allons gérer la partie hors ligne. En effet, si vous essayez aujourd'hui de lancer l'application sans connexion, elle essayera d'appeler l'API et retournera une erreur. La solution est de stocker la valeur du compteur dans une mémoire locale. Mais il faut aussi être capable une fois la connexion retrouvée, d'envoyer les nouveaux clics à notre API. La solution est de créer une nouvelle variable **defer** qui comptera le nombre de clics faits hors ligne. Pour détecter si l'utilisateur est en ligne ou non, nous utilisons la propriété **navigator.isOnline**. Pour garder en mémoire le nombre de clics faits hors ligne et l'ancien total de clics, nous stockerons ces données dans le localStorage.
Ajoutons cette logique à notre code dans public/js/app.js
Le compteur et la valeur defer seront 0 par défaut ou la valeur du localStorage si elle existe.

```
// Par défaut le compteur est à zéro
var count = localStorage.count || 0;
var defer = localStorage.defer || 0;
```

Ensuite, nous déclarons une fonction updateCounter que l'on pourra appeler aussi bien en mode en ligne qu'hors ligne. Si la connexion est rompue, on incrémente la valeur defer, la valeur du compteur devient donc la somme de l'ancien total du compteur retenu avant la perte de connexion avec la valeur defer. Dès que la connexion est retrouvée, nous passons la valeur defer à l'api d'incrément qui augmentera directement la valeur côté serveur du nombre de clics réalisés pendant la perte de connexion.

```
// Cette fonction va automatiquement gérer l'incrément du compteur si nous sommes
// en ligne ou hors ligne
function updateCounter(){
 if(navigator.onLine){
  api('add', defer, function(err, res){
   if(err){return;}
   count = localStorage.count = countElement.innerText = res.counter.count;
   localStorage.defer = defer = 0;
  });
 } else {
  localStorage.defer = defer;
  countElement.innerText = parseInt(count) + parseInt(defer);
 }
}
// On remplace l'appel de l'API au chargement par l'appel à la fonction updateCounter
updateCounter();
```

Enfin sur le clic du bouton principal, on incrémente la valeur defer et on appelle la fonction ci-dessus. Les deux cas (avec ou sans connexion) sont gérés automatiquement.

```
// Au clic sur le bouton d'incrément on ajoute un coeur au compteur
// puis on met à jour le span "count"
loveBtn.addEventListener('click', function() {
 defer++;
 updateCounter();
}, false);
```

Vous pouvez maintenant tester votre application en mode hors ligne. Pensez à couper votre serveur Web mais aussi à vous mettre en mode offline sur votre navigateur. N'oubliez pas de changer le nom de votre cache dans le sw.js pour que votre nouveau code js soit bien pris en compte. Si tout va bien, vous devriez être capable d'incrémenter le compteur aussi bien en mode hors ligne qu'avec une connexion. •

Suite dans le numéro 204

Dites bonjour à **Angular**

Cédric OUTREVILLE
Ingénieur d'études et développement Front chez **SFEIR**

Wassim CHEGHAM
(@manekinekko)
Developer Advocate chez **SFEIR**
et **Google Developer Expert (GDE)**
en Technologies Web

Afin de bien comprendre le principe d'un framework, il est utile de connaître, au moins dans les grandes lignes, son histoire … AngularJs a été créé en 2011 par Miško Hevery et Adam Abronsw. D'abord développé et utilisé en interne, AngularJs a été ouvert au monde open source et s'est trouvé de nombreux utilisateurs et contributeurs. Après le succès d'AngularJs (1.x), la version 2 a donné un nouveau départ au framework en améliorant ses faiblesses.
Ce nouveau départ a eu du mal à être accepté par la communauté à cause d'un lourd remaniement.

Avant l'arrivée des actuels frameworks, nous disposions déjà d'outils tels que Backbone ou jQuery pour nous faciliter l'écriture du JavaScript. Oui, mais nous apportaientils un cadre de développement (très utile dans nos applications d'entreprise), du data binding (uni ou bidirectionnel), un système de routage, un gestionnaire de formulaires ou encore une injection de dépendances, sans oublier tout le nécessaire pour les tests unitaires et E2E ? Ces outils permettaient de résoudre des problème spécialisés mais n'apportaient pas de cadre de développement. Angular Js et Angular nous offre nt ce cadre.
Dans la suite de cet article, nous allons présenter les différents concepts proposés par la nouvelle version d'Angular. A noter que ces derniers, mériteraient un article complet chacun; c'est pour cela que nous allons simplement présenter succinctement quelques concepts sans forcément rentrer dans le détail.

C'est juste "Angular", pas Angular 2 ou Angular 42

Angular va dorénavant respecter le versionnage sémantique X.Y.Z. Ce système permet de donner du sens au numéros des versions :

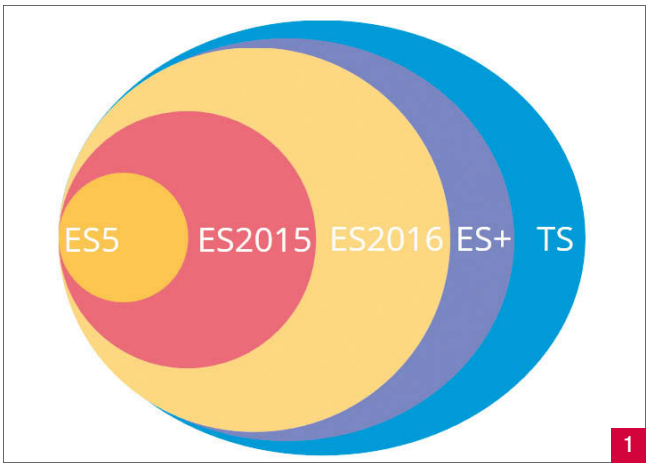
- Z : signifie qu'il y a eu des corrections de bug ;
- Y : signifie qu'il y a eu des nouvelles fonctionnalités (mineures) ;
- X : signifie qu'il y a eu des changements majeurs (potentiel breaking changes).

L'équipe prévoit de sortir une version mineur par mois, et une version majeure tous les 6 mois (pas forcément avec des breaking changes). Il y aura cependant une petite exception : l'équipe pense sauter la version 3 et passer directement à la version 4. La prochaine version sera donc Angular 4 et est prévue pour début Mars 2017.
Mais cela n'est le but de ce dossier, si vous êtes curieux, n'hésitez pas à nous contacter sur Twitter pour plus d'informations et d'anecdotes.

La complexité du monde Web

L'innovation des clients Web

JavaScript, officiellement appelée ECMAScript (ES) est né suite à la "guerre" des navigateurs en Microsoft (IE) et NetScape, pour standardiser le langage et améliorer nos développements Web. De plus en plus supporté par les produits Microsoft, ES connaît actuellement cycle de release accéléré — une version par an — apportant des fonctionnalités de plus en plus riches. Vous trouverez plus de détails sur la page officielle [1].
ES2015 a récemment introduit la notion de Classe [2] et bientôt les décorateurs [3] seront eux aussi ajoutés dans la future version, ES2016 ou probablement ES2017.



1

Microsoft est revenu à la charge en créant cette fois-ci TypeScript. Ce langage est une surcouche de la dernière release d'ES2015. Elle porte un accent sur le côté typage statique du langage. Heureusement pour certains puristes, cela reste un typage optionnel. [Fig.1]
Pour en savoir plus sur TypeScript et le tester directement sur votre navigateur, vous pouvez visiter le site officiel [4].
Quel est le rapport avec Angular ? L'équipe core d'Angular a choisi d'utiliser TypeScript pour développer Angular, justement parce que TypeScript apporte un typage statique qui — ne le nions pas — améliore grandement notre expérience développeurs. Si vous êtes curieux et voudriez savoir pourquoi Angular est développé en TypeScript, Victor Savkin a rédigé un article détaillé à ce sujet [5].

Un framework évolutif et compatible

Le monde du Web a énormément évolué en très peu de temps. Les navigateurs se sont décuplés ; entre Chrome, Safari, Firefox, Opéra ou le remplacement de IE par Edge. Chacun de ces navigateurs utilise une implémentation du moteur JavaScript respectant plus ou moins les standards du TC39. En plus de cette recrudescence, s'ajoute l'apparition des smartphones disposant de leur propre navigateur (pour certains) ainsi que leurs propres contraintes et ergonomie.
Angular a su s'adapter, tout comme d'autres frameworks et librairies,

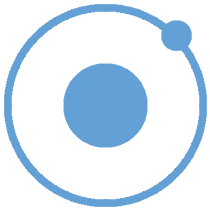
pour nous faciliter la vie et rendre un code unique, compatible pour tous, ce qui améliore énormément sa maintenabilité. Il va sans dire qu'Angular a beaucoup évolué depuis sa création. Son premier objectif était de permettre de développer plus rapidement et plus simplement. Très vite, une question de performance a été soulevée. L'utilisation du DOM n'a pas toujours été optimale. Un nouveau départ a été pris sur la version 2 avec l'amélioration des temps de chargement, l'optimisation de la compilation des templates ainsi que le rendu. La version 2 d'Angular permet même d'être couplée avec d'autres librairies telles que React ou Polymer – voire simplement les Web Components – de manière plus aisée qu'avec AngularJs [6].

Aujourd'hui, Angular est utilisé avec d'autres outils et runtimes de développement multi-plateformes tels que Ionic, Electron ou encore NativeScript. Il est même étendu avec Angular Universal qui permet de prégénérer les rendus de l'application côté serveur.

De manière générale, les objectifs d'Angular sont multiples :

- Viser le développement mobile ;
- Respecter des standards du Web ;
- Améliorer les performances ;
- Diversifier son accessibilité entre ES5, ES2015+, TypeScript et Dart ;
- Être plus simple à écrire et à comprendre ;
- Utiliser des architectures orientées composants ;
- Être universel

Faciliter le multi-plateformes



La forte croissance du Web et son attachement à la compatibilité a inspiré le monde mobile et a donné naissance à plusieurs solutions hybrides. De là, différents frameworks ont été créés afin de soustraire aux développeurs la partie native spécifique à chaque device.

Par exemple, Ionic 2 se base sur Angular pour développer des applications mobiles hybrides. Le concept de ce framework est basé sur l'utilisation d'une WebView, un conteneur capable d'interpréter le JavaScript, HTML, CSS; autrement dit : un navigateur. De ce conteneur, et grâce à Cordova, Ionic donnera une surcouche afin que des plugins développés en natif puissent être déclenchés par l'application Web en JavaScript.

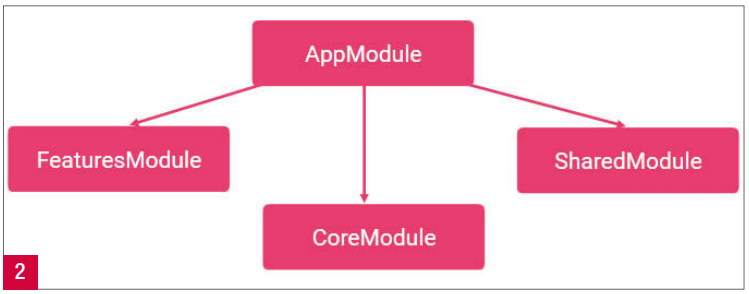
Parmi les nouveautés notables apportées par Angular, nous retrouvons la

notion de Renderers – moteurs de rendu. Ces Renderers nous permettent de s'abstraire de la plateforme sur laquelle tourne l'application Web. Par défaut, et parce que c'est le cas de 99% des applications développées avec Angular, le framework est livré avec un DOM Renderer dédié aux navigateurs. Autrement dit, une application développée avec Angular n'est pas dépendante du DOM, et donc du navigateur.

Cette abstraction liée à la compilation des templates HTML et au rendu permet à Angular de tourner dans des environnements nouveaux. Par exemple, Angular est capable de tourner dans un contexte de Web Workers, permettant d'améliorer considérablement les performances. Tout le processus de compilation des templates a été pensé pour tourner dans un Web Worker; et les compilateurs des futures version d'Angular tourneront par défaut dans ces Web Workers. Il est également possible d'exécuter une application

Angular côté serveur afin de faire du Server-Side Rendering (SSR). Cette fonctionnalité de SSR est rendue possible grâce au module angular-universal. Pour en savoir plus sur Angular Universal, vous pouvez voir ces slides [7].

D'autres projets permettent d'utiliser Angular comme framework pour réaliser nos applications multiplateformes. C'est le cas de NativeScript, proposé par Telerik (Progress). NativeScript est un runtime permettant de développer des applications mobiles "natives" en JavaScript. Contrairement à Ionic, NativeScript produit réellement des applications mobiles natives. Telerik propose donc Angular Native (pour le moment appelé NativeScript for Angular) pour nous aider à réaliser ces applications. Si vous voulez en savoir plus sur NativeScript, voici la dernière vidéo [8] présentée au Angular Connect 2016.



2

Découverte des nouveaux concepts

Une application Angular va s'organiser sous forme d'un arbre de Modules. Par exemple : [Fig.2]

Angular propose de développer des applications modulaires. Un module est responsable de regrouper des composants et services sous forme de "features" ou fonctionnalités. Ainsi, nous pourrons créer plusieurs modules dont chacun est responsable d'une seule et unique fonctionnalité. Nous pouvons qualifier un module comme étant un arbre de composants, comme illustré dans la figure ci-dessus. Nous développerons le concept des composants dans la suite de l'article.

Un module est défini comme suit :

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { HttpModule } from '@angular/http';

import { AppComponent } from './app.component';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    FormsModule,
    HttpModule
  ],
  exports: [HttpModule],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Comme vous pouvez le constater, nous avons déclaré une classe ES2016 laquelle nous avons annoté avec le décorateur **@NgModule()**. Ce décorateur prend la configuration suivante :

- **declarations** : ce tableau contient la liste des Components/Directives et Pipe gérés par ce module ;
- **imports** : la liste des modules externes ;
- **exports**: la liste des dépendances à exporter pour qu'elles puissent être importées par d'autres modules (utile pour les modules enfants) ;
- **providers** : la liste des services Angular ;
- **bootstrap** : la liste des composants à instancier au chargement de l'application. [Fig.3]

Il existe d'autres propriétés que nous pouvons utiliser pour configurer un module, comme **schemas,** utile lors de l'intégration des Custom Elements (Web Components) dans une application Angular, ou encore **entryComponents** lorsque nous voulons faire de la compilation offline, ou Ahead-Of-Time (AOT). Mais ces cas d'usage ne font pas partie du sujet de ce dossier et mériteraient chacun un article complet.

Nous rappelons encore qu'il est fortement recommandé d'organiser son application sous forme de modules spécialisés. Par exemple, nous pouvons regrouper les dépendances techniques comme **BrowserModule**, **FormsModule** et **HttpModule** dans un module nommé **CoreModule**. Puis nous injectons **CoreModule** dans le module principal AppModule. Voici une vue graphique — générée avec l'outil angular2-dependencies-graph [9] — d'un exemple de décomposition par module : [Fig.4]

Notez également que beaucoup de fonctionnalités proposées par Angular 2 ont été séparées dans des modules externes, telles que **FormsModule**, **RouterModule** ou encore **HttpModule**. Mais également lorsque vous utiliserez des librairies externes — les composants Material Design ou Firebase par exemple — ces librairies seront également packagées dans des modules que vous allez importer dans vos modules. Les modules sont les points d'entrées, comme expliqué précédemment.

La syntaxe des templates

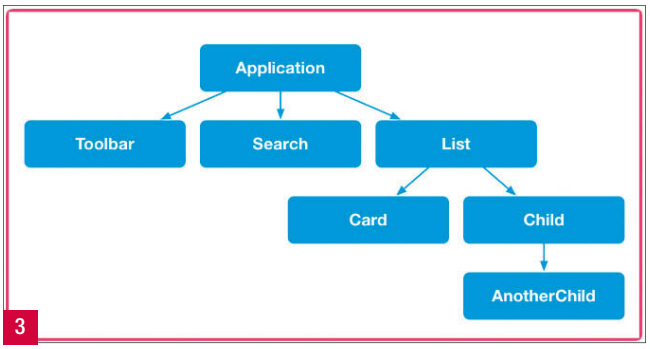
Angular a toujours utilisé le concept du data binding (liaison des données) pour synchroniser le modèle et la vue. Pour rappel, en JavaScript, lorsque nous voulons modifier le rendu dynamiquement, nous devons nou- mêmes relier le code de présentation et le code logique avec des liens (identifiants, ou classes et tous autres sélecteurs CSS) ou recompiler complètement la vue.

Par exemple, pour afficher le nom d'un utilisateur dans une application Web en Vanilla Js, il faudra :

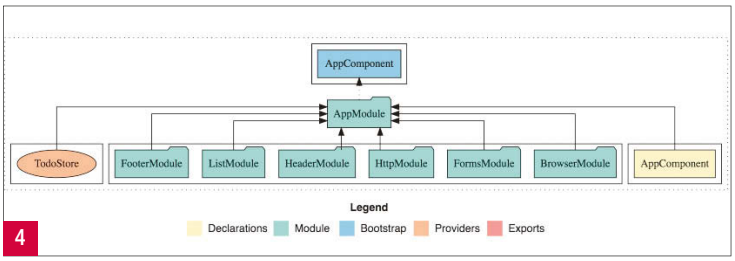
```
<p>Bonjour <span id="username"></span></p>
<script>
  let user = { id : 0, name : "Kara" };
  document.getElementById('username').innerHTML = user.name;
</script>
```

Avec Angular, une simple double accolade suffit. Nous appelons cela une interpolation. Lorsque les propriétés du DOM sont modifiées par le composant ou par interaction avec l'utilisateur (one-way), Angular se charge de remplacer l'expression **{{ expression }}** par sa valeur.

Les bindings sont directement gérés par la syntaxe dans la vue. En effet, les interpolations restent inchangées mais lorsqu'un composant A doit propager un nouvel état vers un composant B, nous allons différencier les syntaxes suivantes :



3



4

- **[property]="etat"** signifie que composant A transmet un nouvel état au composant B ; nous appelons cela : propagation descendante ;
- **(propertyChange)="etat"** signifie que le composant B déclenche un événement appelé "propertyChange" afin de notifier le composant B du nouvel état qu'il souhaite propager. Nous appelons cela une propagation ascendante ;
- **[(property)]="etat"** est un two-way data binding, c'est une combinaison des deux concepts précédents.

Cette différenciation syntaxique permet de comprendre rapidement le code en un minimum de caractères; ainsi nous avons un aperçu des effets désirés et potentiellement leurs conséquences. Nous allons détailler ces concepts plus loin dans la section dédiée aux composants.

Les directives

En Angular, la notion de "directives" est essentiellement réservée aux attributs permettant une interaction entre une partie logique de l'application et le DOM. Autrement-dit, les directives permettent d'étendre le langage HTML avec des attributs de présentation.

Pour créer une directive, il suffit de créer une classe ES2015 et de l'annoter avec un décorateur **@Directive()**. Ce décorateur permet d'ajouter des métadonnées comme la propriété **selector** qui définit comment la directive sera sélectionnée (créée).

Voici un exemple d'une directive :

```
@Directive({
  selector: '[red]'
})
export class RedDirective {}
```

Les directives d'Angular sont réparties en 3 catégories. Découvrons-les.

Les directives structurelles

Les directives structurelles sont celles qui modifient le rendu en effectuant une modification du DOM par ajout ou suppression d'éléments ou de texte. Angular fournit les directives structurelles suivantes : ***ngIf**, ***ngSwitch** et ***ngFor**.

De par l'expression que nous lui donnons, le rendu sera modifié selon un template. L'astérisque * est un sucre syntaxique permettant l'utilisation de

la balise **<template></template>**, proposée par le standard des Web Components. Par exemple :

```
<div *nglf="true"></div>

<!-- equivalent à -->
<template [nglf]="true">
  <div></div>
</template>
```

Les directives d'attribut

Dans le cadre de cette catégorie, la directive va altérer l'apparence ou le comportement d'un élément. Par exemple, **ngClass** qui est une directive fournie par Angular, va permettre d'appliquer une série de classes CSS à l'élément sur lequel elle est instanciée.

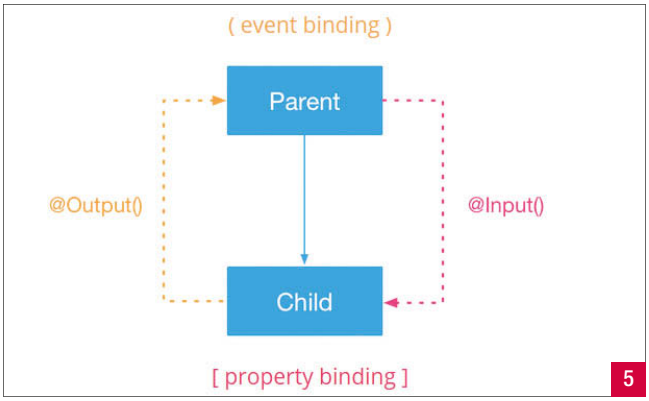
Les composants

Les directives de type "composant" seront certainement celles que nous utiliserons le plus. C'est une directive avec un style CSS et un template HTML. Voici un exemple d'un composant :

```
@Component({
   selector: 'app-toolbar',
   templateUrl: './app.component.html',
   styleUrls: ['./app.component.css']
})
export class AppComponent {}
```

Angular prône la composabilité. C'est-à-dire que notre application Angular sera formée de composants. Chacun de ces composants est responsable de son état interne et n'a pas le droit de modifier directement l'état interne d'un autre composant. Pour cela, Angular nous propose (sans l'imposer) une approche basée sur une propagation de flux unidirectionnelle — **Unidirectional Dataflow**.
Chaque composant (et par définition chaque directive) en Angular possède une sorte d'interface d'entrée/sortie. Un composant peut exposer des propriétés permettant à d'autres composants de lui passer des informations et de propager ainsi un changement d'état. Pour cela, le composant doit annoter sa propriété avec le décorateur **@Input()**.
De la même manière, un composant peut également exposer un événement que d'autres composants peuvent écouter afin de réagir et déclencher potentiellement d'autres actions. Pour cela, la propriété de classe du composant doit être annotée avec le décorateur **@Output()**. [Fig.5]
Voyons cela avec un exemple de code.
Prenons par exemple un fichier **app.module.ts** contenant la déclaration d'un module comme vu précédemment. Nous allons y ajouter un composant **DireBonjourComponent** :

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { HttpModule } from '@angular/http';

import { AppComponent } from './app.component';
import { DireBonjourComponent } from './components/dire-bonjour/dire-bonjour.
component';

@NgModule({
```



5

```
 declarations: [
  AppComponent,
  DireBonjourComponent
 ],
 imports: [
  BrowserModule,
  FormsModule,
  HttpModule
 ],
 bootstrap: [AppComponent]
})
export class AppModule { }
```

Ensuite, dans **app.component.ts** nous déclarons une fonction **afficher()** à exécuter lors du déclenchement d'un évènement "**alert**" levé par le composant **DireBonjourComponent** :

```
import { Component } from '@angular/core';
import { DireBonjourComponent } from './dire-bonjour';

@Component({
 selector: 'app-root',
 templateUrl: 'app.component.html',
 styleUrls: ['app.component.css']
})
export class AppComponent {
 title = 'Hello world ? :)';

 afficher(content: string) {
    console.log(content);
 }

}
```

Dans le template de notre **AppComponent**, nous devons utiliser les crochets "**[]**" pour passer l'information au composant via sa propriété **nom**, puis nous utilisons les "**()**" pour écouter son évènement **alert**.

```
<app-dire-bonjour [nom]="'Brad'" (alert)="afficher($event)"></app-dire-bonjour>
```

Passons au composant enfant qui doit disposer d'une propriété d'input et une autre capable d'émettre un événement.

```
import { Component, Input, Output, EventEmitter } from '@angular/core';
```

```
@Component({
 selector: 'app-dire-bonjour',
 templateUrl: 'dire-bonjour.component.html',
 styleUrls: ['dire-bonjour.component.css']
})
export class DireBonjourComponent {

 @Input() nom: string;
 @Output() alert: EventEmitter<string>;

 constructor() {
    this.alert = new EventEmitter<string>();
 }

 clicked(nom: string) {
    this.alert.emit(nom);
 }

}
```

Et enfin, le template **dire-bonjour.component.html** :

```
Bonjour <span (click)="clicked(nom)">{{ nom }}</span>
```

En résumé :

- **[nom]** est l'input défini dans **dire-bonjour.component.ts** qui prendra pour valeur la chaine "Brad" ;
- **(alert)** est l'évènement également défini dans **dire-bonjour.component.ts** et exécutera **afficher($event)** où **$event** sera : **Brad** ;
- Lors d'un click sur le texte "Brad", l'évènement **click** appellera la méthode **clicked**, qui, elle-même, émettra l'évènement **alert**.

Les injections de dépendances

AngularJs était le premier framework front à avoir introduit la notion d'injection de dépendances (DI). Ce concept, dont nous connaissons tous les avantages, permet en réalité d'augmenter la compréhension et la modularité d'un projet. Pour cela, il est très recommandé que la logique métier soit implémentée dans les services afin de bénéficier de cette DI. Après tout, Angular reste un framework orienté services.

En Angular, ce concept de DI a été complètement revu et amélioré. Par exemple, les services sont maintenant de simples classes ES2015 annotées avec le décorateur **@Injectable()**. Fini les différentes notions de const(), value(), service() ou factory() d'AngularJs.

Prenons l'exemple de la fameuse "Todo List" qui nous permet de créer une liste de tâches à effectuer. Ces tâches sont bien évidemment stockées côté serveur. En abstrayant dans un service l'utilisation du service **Http** d'Angular, nous pourrons plus tard, changer cette implémentation en créant simplement une nouvelle classe qui respecte la même interface. Grâce à la DI, nous pouvons changer à souhait l'implémentation à utiliser sans impacter le reste de l'application. Aussi, nous pouvons créer un service bouchonné que nous pouvons utiliser dans nos tests unitaires. Ce service peut également être utilisé pendant la phase de prototypage en attendant que — par exemple — l'équipe back termine l'implémentation du service REST. La liste des exemples et cas d'usages est longue...

Regardons maintenant comment Angular nous permet de bénéficier de la DI. Nous allons faire le choix de créer une classe abstraite que nous allons utiliser comme interface. Ou une simple interface aurait suffi à vrai dire. Nous allons nommer cette classe **AbstractApiService** dans un fichier **api.service.ts** — en respectant les conventions de nommage [12] :

```
import { Observable } from 'rxjs/Observable';

export abstract class AbstractApiService {
   getPosts: () => Observable<string[]>;
}
```

Nous pouvons désormais créer une classe **MockApiService** qui implémente cette classe abstraite :

```
import { Injectable } from '@angular/core';
import { Http } from '@angular/http';
import { Observable } from 'rxjs/Observable';
import 'rxjs/add/operator/map';

import { ApiService } from './api.service';
const BASE_URL: string = 'https://jsonplaceholder.typicode.com';

@Injectable()
export class MockApiService implements AbstractApiService {

 constructor(public foo: FooService) {}

 getPosts(): Observable<string[]> {
  return {};
 }
}
```

Vous remarquez que nous avons annoté la classe **MockApiService** avec **@Injectable()**. Ce décorateur indique à Angular d'injecter les dépendances — éventuellement — réclamées par ce service **MockApiService**; dans ce cas là **FooService**. Sans ce décorateur, Angular vous afficherait une erreur au runtime. Il est donc recommandé de **toujours** annoter les services Angular avec ce décorateur, comme l'a toujours dit John Papa :). Utilisons maintenant **MockApiService**. Pour cela, nous allons faire le choix d'injecter ce service localement au niveau du composant :

```
import { Component, OnInit } from '@angular/core';
import { ApiService, MockApiService } from './shared';

@Component({
 selector: 'app-root',
 templateUrl: 'app.component.html',
 providers: [{
    provide: ApiService,
    useClass: MockApiService
 }],
 styleUrls: ['app.component.css']
})
export class AppComponent implements OnInit {

 title = 'Hello world ? :)';
 posts: Array<any> = [];

 constructor (private api: ApiService) {}
```

```
  ngOnInit () {
    this.api.getPosts()
      .subscribe((apiResponse) => this.posts = apiResponse);
  }

  alert(content: string) {
    console.log(content);
  }
}
```

Ainsi, le template de notre composant peut afficher la liste :

```
<h1>{{title}}</h1>
<app-dire-bonjour [nom]="'Igor'" (clique)="alert($event)"></app-dire-bonjour>
<div>
  <p> ma liste de poste via mon apiService mocked</p>
  <ul>
  <li *ngFor="let post of posts">
      <h1>{{ post.title }}</h1>
      <p>{{ post.body }}</p>
  </li>
  </ul>
</div>
```

L'injection de dépendances est réalisée grâce à la syntaxe canonique **{ provide: ApiService, useClass: MockApiService }**. Cette syntaxe permet en quelque sorte de configurer l'injecteur de dépendances en lui indiquant que lorsqu'un service demande **ApiService**, l'injecteur doit lui fournir une instance de la classe **MockApiService**.
Heureusement, il existe une syntaxe plus courte pour les cas exceptionnels. Par exemple, si un service demande la classe **MockApiService**, nous pourrions écrire ce raccourci :

```
providers: [{
  provide: MockApiService,
  useClass: MockApiService
}],

// est equivalent à
providers: [MockApiService],
```

Sachez qu'il est également possible d'injecter ces services de manière globale, au niveau d'un **NgModule**. Dans ce second cas, l'instance de ce service sera la même partout dans ce module. Nous parlons dans ce cas de **Singleton**. Un autre exemple aussi intéressant : dans une application hybride ou multiplateforme, il est possible d'avoir deux implémentations différentes pour utiliser une même fonctionnalité native. Dans ce cas, nous utiliserons **useFactory** qui permet de configurer l'instance à créer. Par exemple :

```
{
  provide: MyHttpService,
  useFactory: (dep1, dep2) => {

    if (IS_NODE) {
      return new NodeHttp(dep1, dep2);
    }
    else if (IS_BROWSER) {
      return new Http(dep1, dep2);
    }

  },
  deps: [dep1, dep2]
}
```

Il existe aussi d'autres syntaxes pour configurer les injecteurs en Angular mais cela nécessiterait un article complet sur le sujet tellement l'API est riche et puissante.

Form et validation

Les formulaires en Angular ont été enrichis et sont encore plus simples à utiliser. Certes, à l'instar d'AngularJs, il est toujours possible de déclarer et utiliser les formulaires directement dans le template HTML d'un composant. Mais Angular ajoute une nouvelle API qui privilégie plus une approche programmatique. Nous parlons de Template Driven pour la première, et Model Driven pour la seconde.

Template Driven Forms

Le module permettant d'utiliser les Template Driven Forms s'appelle **FormsModule**.
Dans l'exemple suivant, nous allons surcharger le template afin de créer le formulaire à traiter, de valider les champs et d'afficher les messages d'erreurs si la saisie ne correspond pas aux spécifications. Afin d'illustrer cette approche, nous allons créer un composant avec pour seule responsabilité de remonter la donnée d'un champ de saisie au composant parent :

```
import { Component, Output, EventEmitter } from '@angular/core';

@Component({
  selector: 'app-template-driven-form',
  templateUrl: './template-driven-form.component.html'
})
export class TemplateDrivenFormComponent {

   name: string = '';
  @Output() submitted: EventEmitter<string>;

  constructor() {
     this.submitted = new EventEmitter<string>();
  }


   send(value: string) {
      this.submitted.emit(value.name);
   }

}
```

Le template contient tout le nécessaire gérer le formulaire ainsi que sa validation :

```
<p>Voulez-vous changer de nom ?</p>
<form #userForm="ngForm" (ngSubmit)="send(userForm.value)">
```

```
  <label>Nouveau nom :</label>
  <input type="text" name="name" ngModel required>
  <div *ngIf="name.errors?.required">
  <span>Je suis sur que vous avez un prénom</span>
  </div>

  <input type="submit" value="Changer de nom">
</form>
```

Maintenant, utilisons ce formulaire dans le composant principal :

```
import { Component } from '@angular/core';

@Component({
 selector: 'app-root',
 templateUrl: 'app.component.html'
})
export class AppComponent {

 title:string = 'Hello world ? :)';
 name: string = 'Misko';

 constructor () {}

 alert(content: string) {
  console.log(content);
 }

 changeName(name: string) {
    this.name = name;
 }

}
```

Et son template :

```
<h1>{{title}}</h1>
<div>
  <app-dire-bonjour [nom]="name" (clique)="alert($event)"></app-dire-bonjour>
  <app-template-driven-form (submitted)="changeName($event)"></app-template
-driven-form>
</div>
```

Avec les Template Driven Forms, tout est présent dans le HTML. Nous aurons accès à la valeur de retour de ce formulaire depuis sa propriété **userForm.value** quicontient tous les champs du formulaire — également appelés **Form Controls** — sous forme d'objet JSON. Cette approche est très similaire à celle introduite par AngularJs, à part la syntaxe qui diffère; mais le concept est le même.

Model Driven Forms

L'usage d'un Model Driven Forms se fait avec le module **ReactiveFormsModule**.

Avec cette approche, tout est défini dans la classe du composant. Nous utilisons ensuite l'API **FormGroup** et **formControlName** — entre autres —pour associer le code du formulaire avec son template HTML.

Cette approche offre quelques avantages. Par exemple, elle nous permet de tester unitairement le formulaire puisque tout le formulaire est créé programmatiquement et nous n'aurons pas besoin de charger l'HTML pour cela. Les Template Driven Forms quant à eux, peuvent toujours être testés grâce aux tests E2E.

Regardons maintenant comment implémenter un formulaire ainsi que de la validation avec l'approche Model Driven Forms. Après avoir ajouté le module **ReactiveFormsModule** dans les imports du module courant, nous allons créer le composant suivant :

```
import { Component, OnInit, Output, EventEmitter } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';

@Component({
 selector: 'app-model-driven-form',
 templateUrl: './model-driven-form.component.html'
})
export class ModelDrivenFormComponent implements OnInit {

 form: FormGroup;
 @Output() submitted: EventEmitter<string>;

 constructor(private formBuilder: FormBuilder) {
    this.submitted = new EventEmitter<string>();
 }

 ngOnInit() {
    this.form = this.formBuilder.group({
  name: [ 'Brad', [Validators.required] ]
    });
 }

 send() {
    this.submitted.emit(this.form.controls['name'].value);
 }
}
```

Alors que le template sera allégé :

```
<p>Voulez-vous changez de nom ?</p>
<form [formGroup]="form" (ngSubmit)="send(form)">
   <label>Nouveau nom:</label>
   <input type="text" formControlName="name">
   <p *ngIf="form.controls.name.errors">Je suis sûr que vous avez un prénom</p>

  <button type="submit">Changer de nom</button>
</form>
```

Le fichier **app.component.ts** — ainsi que son template HTML — ne sera pas modifié par rapport à notre exemple précédent sur les Template Driven Forms.

Vous remarquez qu'avec cette approche, tout se passe dans la classe. Pour un simple champ de saisie avec une simple validation **required**, cela est trivial. Mais lorsque les champs se multiplient et les validations se complexifient, le choix du Model Driven Forms prend tout son sens. Nous pouvons même imaginer générer dynamiquement un formulaire en fonction d'instructions envoyées par le serveur : schéma de données, règles de validation, libellés, etc.

La gestion des routes

Le routage d'une application Angular est aussi simple que puissant. Il a été revu et amélioré par rapport au routeur d'AngularJs. Voyons cela.
Nous commençons par importer le module **RouterModule** dans notre module applicatif. Ensuite, nous configurons les routes de ce module. Pour cela nous allons ajouter un fichier **app.routes.ts** dans lequel nous allons utiliser la méthode **RouterModule.forRoot()** pour créer et enregistrer ces routes :

```
import { RouterModule } from '@angular/router';
import { AppComponent } from './app.component';
import { ModelDrivenFormComponent } from './components/model-driven-form/
model-driven-form.component';
import { PostsComponent } from './components/posts/posts.component';

export let AppRoutes = RouterModule.forRoot([
 { path: '', component: ModelDrivenFormComponent },
 { path: 'compte', component: ModelDrivenFormComponent },
 { path: 'posts', component: PostsComponent }
]);
```

Comme vous pouvez le constater, le routeur d'Angular est un routeur de composants. C'est-à-dire qu'il gère le routage au niveau des composants et non pas des vues – comme le routeur par défaut d'AngularJs.
Il suffit ensuite d'importer la déclaration **AppRoutes** dans **app.module.ts** :

```
...
import { AppRoutes } from './app.routes';
...
imports: [
  BrowserModule,
  ReactiveFormsModule,
  HttpModule,
  AppRoutes
 ],
...
```

A ce stade, les règles de routage sont enregistrées dans le module.
Regardons maintenant comment la navigation fonctionne réellement au sein de l'application :

```
<h1>{{title}}</h1>
<div>
  <app-dire-bonjour [nom]="name" (clique)="alert($event)"></app-dire-bonjour>
  <ul>
    <li><a [routerLink]="/compte/"]>Changer de nom</a></li>
    <li><a [routerLink]="/posts/"]>Posts de l'api</a></li>
  </ul>

  <router-outlet></router-outlet>

</div>
```

Lorsque nous naviguons dans l'application en cliquant par exemple sur un lien, la directive **routerLink** sert à indiquer au routeur le chemin – ou la prochaine route – à activer. Le routeur récupère ensuite le composant associé à cette route puis l'instancie. Une fois créé, le contenu HTML du composant s'affiche dans la ou les balise(s) **router-outlet**, spécifiées dans la configuration (nous l'avons pas fait dans notre configuration car ce paramètre est optionnel).
Le Routeur d'Angular est également capable de charger ou précharger des modules de manière tardive, ce que nous appelons dans le jargon : le **Lazy Loading** !
Pour indiquer au routeur qu'il doit charger un module en "lazy loading", il suffit de lui indiquer la configuration suivante :

```
import { RouterModule } from '@angular/router';
import { AppComponent } from './app.component';
import { ModelDrivenFormComponent } from './components/model-driven-form/
model-driven-form.component';
import { PostsComponent } from './components/posts/posts.component';

export let AppRoutes = RouterModule.forRoot([
 { path: 'posts', loadChildren: 'app/posts/posts.module#PostsModule' }
]);
```

C'est extrêmement simple. La propriété **loadChildren** indique au routeur que le module **PostsModule** doit être chargé et créé uniquement lorsque la route "**/posts**" est activée. Ceci permet donc d'alléger le code chargé au lancement de l'application et d'accélérer ainsi son temps de chargement.
A noter que pour que le lazy loading fonctionne il faudrait que le module **PostsModule** soit déclaré dans un fichier – bundle – séparé afin qu'il soit chargé séparément. Bonne nouvelle, si vous utilisez le CLI d'Angular, ce travail est fait pour vous !

Formater les données avec les Pipes

A l'instar des filtres d'AngularJS, Angular offre les "Pipes" pour formater les données affichées dans la vue. Par exemple : **{{ 'programmez' | uppercase }}** formatera l'affichage en **PROGRAMMEZ**.
Il nous est bien évidement possible de créer nos propres Pipe. De plus, l'API est on ne peut plus simple :

```
import {Pipe, PipeTransform} from '@angular/core';

@Pipe({
 name: 'trim'
})
export class TrimPipe implements PipeTransform {
 transform(value: any) {
  if (!value) {
   return '';
  }
  return value.trim();
 }
}
```

Côté vue :

```
<div>{{ '  Programmez!      ' | trim }}</div>
```

Dans cet exemple, nous avons créé un Pipe nommé **trim**, qui permet de retirer les espaces au début et à la fin d'une chaîne de caractères.
Dans la terminologie Angular, le Pipe **trim** que nous avons créé peut être

qualifié de **pure**, puisqu'il retourne toujours le même résultat lorsqu'il est appelé avec les mêmes arguments; Et surtout, il ne provoque aucun effet de bord.
Jusque-là rien de nouveau par rapport aux filtres d'AgularJS me diriez-vous. Il se trouve qu'Angular nous permet de faire beaucoup plus avec les Pipes.
Contrairement aux filtres, les Pipes peuvent réaliser des opérations asynchrones. Dans ce cas, ils sont qualifiés d'**impurs**. Les cas d'usages sont nombreux : vous pouvez par exemple vouloir récupérer des informations complémentaires depuis le serveur avant de formater une valeur, ou encore effectuer un traitement asynchrone sur une valeur — dans un Web Worker par exemple. Voyons un exemple — à titre indicatif — de l'utilisation d'un Pipe impur :

```
import {Pipe, PipeTransform} from '@angular/core';
@Pipe({
  name: 'fetch',
  pure: false
})
export class FetchJsonPipe {
  private fetchedValue:any;
  private fetchPromise:Promise<any>;

  transform(value:string, args:string[]):any {
    this.fetchPromise = window.fetch(value)
      .then((result:any) => result.json())
      .then((json:any)   => this.fetchedValue = json);

    return this.fetchedValue;
  }
}
```

Nous avons créé un Pipe **fetch** impur — notez la propriété **{pure: false}**. Ce Pipe prend en paramètre le nom d'un fichier puis exécute une requête réseau et affiche le résultat retourné, comme ceci :

```
<div>{{'heroes.json' | fetch | json}}</div>
```

HTTP

Une fois de plus, Angular propose une API **Http** permettant d'effectuer des appels HTTP. Cette API est fournie par le module **HttpModule**. La grande nouveauté de l'API **Http** par rapport au service **$http** d'AngularJs est qu'elle ne se base plus sur les Promesses mais plutôt sur les **Observables**.
Les Observables vont permettre de supprimer un grand nombre de problèmes connus avec les appels asynchrones ; ils vont aussi améliorer les performances des appels. En effet, avec les Promesses, il n'est pas possible d'annuler un appel ou de gérer le retour successif des données. Alors que les Observables le supportent.
Les Observables font beaucoup débat en ce moment, car certains n'y voient pas d'intérêt et d'autres les évangélisent à tout va. Tout ce que nous pouvons dire c'est que les Observables sont déjà en Stage 1 de la standardisation [10] et nous allons probablement les retrouver dans ES2017. Pascal a rédigé un excellent article sur le sujet [11] que nous vous conseillons de lire. Complétons maintenant notre exemple en utilisant le module **Http**. D'abord, complétons l'interface afin d'augmenter les fonctionnalités qu'elle doit proposer :

```
import { Observable } from 'rxjs/Observable';

export abstract class ApiService {
  getPosts: () => Observable<any>;
  postPost: (body: any) => Observable<any>;
}
```

Implémentons maintenant notre ApiService qui ne correspond plus à l'implémentation de son interface :

```
import { Injectable } from '@angular/core';
import { Http, RequestOptions, Headers } from '@angular/http';
import { Observable } from 'rxjs/Observable';
import 'rxjs/add/operator/map';

import { ApiService } from './';

const BASE_URL: string = 'https://jsonplaceholder.typicode.com';

@Injectable()
export class MockApiService implements ApiService {

  constructor(private http: Http) {}

  getPosts(): Observable<any> {
   return this.http.get(`${BASE_URL}/posts`)
        .map( res => res.json() );
  }

  postPost(body: any): Observable<any> {
     return this.http.post(`${BASE_URL}/posts`, body)
          .map( res => res.json() );
  }

}
```

Simple et efficace, n'est-ce-pas ?

Une convention d'architecture

Contrairement à AngularJS, Angular vient dès sa sortie avec une convention de bonnes pratiques et de recommandations quant à la structure et l'architecture de nos applications. Vous pouvez trouver la totalité de ces règles dans la documentation officielle [12].

Angular CLI

JavaScript devient de plus en plus complexe car les usages du Web deviennent de plus en plus riches. Les utilisateurs sont maintenant plus exigeants. Nos applications Web actuelles et futures ont besoin d'une architecture adaptée à ces nouveaux usages. Une nouvelle expression a vu le jour permettant de décrire ce que ressentent les développeurs : JavaScript-Fatigue [13]. Pour contrer cette lourdeur qui va plus loin que créer un **index.html** avec une balise **<script>**, des starters — ou kit de démarrage — ont été mis en place. Il en existe pour tous les frameworks Web modernes. Pour Angular, il existe des Starters qui permettent d'avoir une base d'une application, avec des exemples de composants, directives, services, … Le starter de Patrick d'AngularClass est l'un des plus utilisés (même pour AngularJs). La spécificité de ce starter est qu'il utilise

Webpack comme bundler [14]. Cependant, ces starters ne font qu'apporter une base pour bien démarrer. Pour la plupart, ils ne permettent pas de nous accompagner tout au long de notre développement. Angular propose son CLI officiel [15]. Il permet bien évidemment de générer une application blanche en une seule commande : **ng new programmez**. Il nous permet également de générer au fur et à mesure tous nos composants, pipes, modules ainsi que les squelettes des fichiers de tests qui vont avec : **ng generate component FooBar**. Il nous aide à packager l'application pour l'intégration et la production : **ng build --aot**. Il est également capable d'exécuter les tests unitaires et E2E : **ng test**. Plus besoin de mettre les mains dans la configuration de Karma ! Le CLI a également pour objectif de renforcer le respect des bonnes pratiques Angular, ce qui est très important pour les projets impliquant plusieurs équipes.

CONCLUSION

Angular est le fruit de la collaboration avec d'autres frameworks et librairies front telles que React ou Ember. Sans parler des équipes TypeScript, Ionic ou encore Telerik. Il propose une architecture orientée composants qui rend le code plus maintenable et réutilisable. Son utilisation de plus en plus populaire donne accès à une communauté dont la présence est forte et active. Depuis son ouverture à l'open source, Angular n'a pas cessé d'évoluer, de disposer de nouveaux modules, d'être utilisé par d'autres frameworks ou de rentrer dans le monde back avec, par exemple, Angular Universal ; sans parler de son utilisation par Ionic pour le développement d'applications hybrides ou encore par NativeScript pour les applications mobiles natives.

Angular n'est plus un framework, c'est une plateforme à part entière. Cependant, grâce à son aspect modulaire, il reste suffisamment très léger et agréable à utiliser. A vous de jouer maintenant en codant votre première application avec Angular et dites-nous ce que vous en pensez. Utilisez le hashtag **#ProgrammezWelcomeAngular** (mettez également nos comptes Twitter **@manekinekko** dans votre tweet pour que nous soyons notifiés). Nous serions ravis de répondre à toutes vos questions et de vous aider à bien démarrer avec Angular.

Liens

[0] http://semver.org
[1] http://bit.ly/arrayfilter
[2] https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Classes
[3] https://github.com/tc39/proposal-decorators
[4] http://www.typescriptlang.org/play/
[5] https://vsavkin.com/writing-angular-2-in-typescript-1fa77c78d8e8#.f4uerqr3t
[6] http://bit.ly/angularwc
[7] http://slides.com/wassimchegham/angular2-universal
[8] https://www.youtube.com/watch?v=aWM-_majn0E
[9] http://bit.ly/angularngd
[10] https://github.com/tc39/proposal-observable
[11] http://bit.ly/ngobservables
[12] https://angular.io/docs/ts/latest/guide/style-guide.html
[13] https://medium.com/@ericclemmons/javascript-fatigue-48d4011b6fc4#.sf9q2lnlj
[14] https://github.com/AngularClass/angular2-Webpack-starter
[15] https://github.com/angular/angular-cli

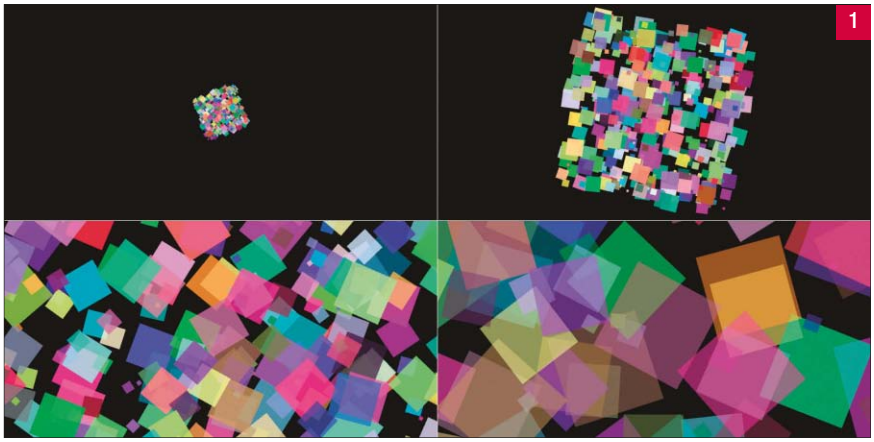
Resources Angular

angular.io - Official Angular site
gitter.im/angular/angular - Angular Community Chat
github.com/angular/angular - Official Angular repo
github.com/angular/universal - Official "Angular Universal" repo
github.com/angular/angular-cli - Official CLI repo

Exploser des pixels avec WebGL et JavaScript

Partie 1

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


1

En s'appuyant sur les shaders – le vertex shader (VS) et le fragment shader (FS) – de WebGL, il s'agit de réaliser une animation de « pixels » s'éloignant d'un point d'origine en tournoyant (autour de leurs centres et autour du centre de l'écran), en grossissant et devenant toujours plus transparents jusqu'à disparaître de l'écran, comme sur la figure suivante qui reprend quatre étapes : [1]

WebGL est simple à utiliser. La difficulté réside dans un dilemme auquel le développeur est inévitablement confronté dès qu'il s'attaque à des effets un peu compliqués :

- Déporter la transformation des points dans les shaders, si bien qu'il suffit d'appeler une fois drawElements () pour rendre tous les éléments lors d'une étape de l'animation, mais c'est au prix d'une inflation du buffer des points et d'une mise à jour fréquente de ce dernier ;
- Déporter la transformation des points dans le programme principal (ie : le programme JavaScript), si bien qu'il est possible de créer une fois pour toutes le buffer des points, mais c'est au prix d'autant d'appels à drawElements () qu'il y a d'éléments à rendre lors d'une étape de l'animation.

Pour être complet, il faut évoquer une troisième solution qui consiste à déporter la transformation des points dans le programme principal, lequel met à jour le buffer des points avec les résultats de cette transformation avant d'appeler une seule fois drawElements (), et ce, à chaque étape de l'animation. Toutefois, cette solution doit être écartée, car elle revient à réduire le VS au rang de passe-plat entre le programme principal et le FS, puisque le VS est alimenté avec des points déjà transformés. Or si le GPU peut assurer la transformation des points, autant l'utiliser, car cela libère le CPU pour d'autres tâches.

Cet article est le premier d'une série de deux. Il est consacré à la logique générale du programme et à l'initialisation de WebGL incluant l'écriture du code des shaders. Le second article sera consacré à la boucle d'animation. Les deux articles ne rentreront pas dans le détail d'un commentaire ligne par ligne du code de l'exemple mis à disposition, mais rien de ce qui concerne WebGL ne sera ignoré.

La solution

L'expérience montre qu'en l'espèce, multiplier les appels à drawElements () pénalise beaucoup les performances. Il faut donc adopter la solution consistant à déporter dans les shaders tous les calculs requis pour transformer et rendre tous les « pixels » à l'occasion d'un unique appel à drawElements () par étape de l'animation.

Cela a deux conséquences :

- Le code du VS est un peu plus compliqué. En effet, le VS ne sert plus seulement de passe-plat entre le programme principal qui transformerait les points, et le FS qui afficherait les pixels comme nous l'avons vu en étudiant comment utiliser WebGL pour un rendu vectoriel : il doit transformer les points.
- Le VS a besoin de toutes les données requises pour effectuer les transformations des points d'un « pixel ». Toutefois, comme il n'a accès qu'aux données spécifiques au point courant (les données passées via les arguments valent pour tous les points, étant fixées avant l'unique appel à drawElements ()), il faut annexer ces données à celles de chaque point du « pixel » dans le buffer des points.

Le code JavaScript

Le programme est structuré en plusieurs parties :

- L'initialisation générale assurée par run () ;
- L'initialisation de l'explosion assurée par explode () ;
- Le rendu d'une étape de l'explosion assurée par nextStep ().

Cet article porte sur la première partie, l'initialisation générale. Un second article présentera les deux autres.

Tout d'abord, il faut créer un élément HTML canvas et obtenir son contexte WebGL, rajouté au document comme enfant d'un élément HTML quelconque (en l'occurrence un élément d'identifiant tagCanvas) :

```
var CANVAS_WIDTH = 800;
var CANVAS_HEIGHT = 600;
var canvas;
var gl;

canvas = document.createElement ("canvas");
canvas.width = CANVAS_WIDTH;
canvas.height = CANVAS_HEIGHT;
document.getElementById ("tagCanvas").appendChild (canvas);
gl = canvas.getContext ("webgl"); // Utiliser "experimental-webgl" pour Internet Explorer
```

Noter que l'élément HTML canvas peut parfaitement avoir été déjà directement mentionné dans le code HTML, auquel cas il n'est pas nécessaire de le créer :

```
<canvas id="tagCanvas" width="800" height="600"/>
```

Ensuite, il faut créer le VS et le FS :

```
var shaderV, shaderF;

shaderV = gl.createShader (gl.VERTEX_SHADER);
gl.shaderSource (shaderV, "attribute vec2 A_xy; attribute vec3 A_rgb; attribute vec4 A_bzdxdy; uniform mat4 U_mW; varying lowp vec3 V_rgb; void main (void) { mat4 m; m = mat4 (1.0); m[0][0] = cos (A_bzdxdy.x) * A_bzdxdy.y; m[1][0] = - sin (A_bzdxdy.x) * A_bzdxdy.y; m[0][1] = sin (A_bzdxdy.x) * A_bzdxdy.y; m[1][1] = cos (A_bzdxdy.x) * A_bzdxdy.y; m[3][0] = A_bzdxdy.z; m[3][1] = A_bzdxdy.w; gl_Position = U_mW * m * vec4 (A_xy.xy, 0.0, 1.0); V_rgb = A_rgb; }");
gl.compileShader (shaderV);
if (!gl.getShaderParameter (shaderV, gl.COMPILE_STATUS))
  alert (gl.getShaderInfoLog (shaderV));
shaderF = gl.createShader (gl.FRAGMENT_SHADER);
gl.shaderSource (shaderF, "varying lowp vec3 V_rgb; uniform lowp float U_a; void main (void) { gl_FragColor = vec4 (V_rgb, U_a); }");
gl.compileShader (shaderF);
if (!gl.getShaderParameter (shaderF, gl.COMPILE_STATUS))
  alert (gl.getShaderInfoLog (shaderF));
program = gl.createProgram ();
gl.attachShader (program, shaderV);
gl.attachShader (program, shaderF);
gl.linkProgram (program);
gl.useProgram (program);
```

Enfin, il faut obtenir des références sur les attributs et les uniformes des shaders afin d'en préciser les valeurs ultérieurement :

```
var A_xy, A_rgb, A_bzdxdy, U_mW, U_a;

A_xy = gl.getAttribLocation (program, "A_xy");
A_rgb = gl.getAttribLocation (program, "A_rgb");
A_bzdxdy = gl.getAttribLocation (program, "A_bzdxdy");
U_mW = gl.getUniformLocation (program, "U_mW");
U_a = gl.getUniformLocation (program, "U_a");
```

La logique

La création du canvas et la récupération du contexte WebGL ainsi que la récupération de références sur les attributs et les uniformes (dont le rôle sera précisé plus loin) ne prêtent pas à commentaires. Le vrai sujet est la création des shaders, et plus précisément l'écriture du code de ces derniers. Avant de plonger dans le code du VS, il faut expliquer ce qui est attendu de ce dernier.

Ce qui est attendu du VS

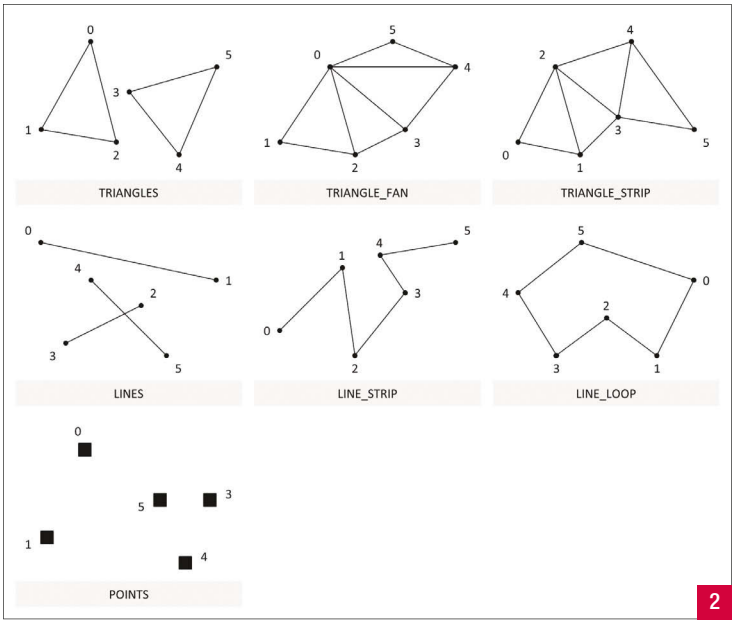
WebGL peut manipuler différents types de primitives : [2]

En l'espèce, chaque « pixel » est représenté sous la forme de deux triangles distincts, c'est-à-dire deux primitives TRIANGLES.

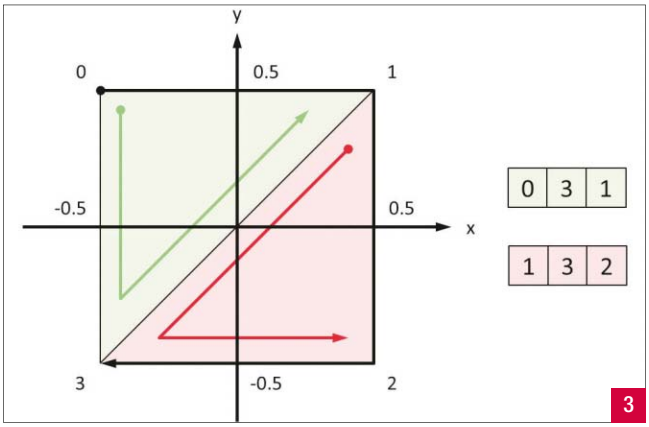
A priori, ce choix n'est pas judicieux.

Pourquoi ne pas utiliser des POINTS ? Parce qu'un point n'est pas une surface, ce qui signifie qu'il est impossible de lui appliquer une transformation telle que la rotation envisagée – le zoom est possible, en jouant sur la variable système du VS gl_PointSize. Va donc pour des triangles.

Mais alors, pourquoi ne pas utiliser des TRIANGLE_STRIP ou des TRIANGLE_FAN ? En effet, ces primitives ne permettent-elles pas de décrire



2



3

un « pixel » en fournissant simplement quatre points ? Utiliser TRIANGLES contraint à fournir deux fois trois points pour chaque triangle, soit six points, dont deux se répètent puisqu'ils sont communs aux deux triangles. Certes, mais c'est inévitable et il est possible de limiter le nombre de points :

- C'est inévitable, car on souhaite transformer et rendre en une seule fois tous les « pixels ». Dans ces conditions, utiliser des TRIANGLE_STRIP et des TRIANGLE_FAN signifie qu'il faut que WebGL permette de transformer et de rendre en une seule fois une de ces primitives par « pixel ». Or WebGL ne le permet pas : c'est un TRIANGLE_STRIP ou un TRIANGLE_FAN par appel à drawArrays () ou drawElements ().
- Il est possible de limiter le nombre de points, car WebGL permet de commander la transformation et le rendu via drawElements () plutôt que drawArrays (). Avec drawElements (), il suffit de fournir quatre points et six indices, trois indices consécutifs référençant les points qui forment un TRIANGLES.

Les deux TRIANGLES sont décrits à l'aide de quatre points formant un carré de 1.0 x 1.0 centré sur (0.0, 0.0), et de deux jeux de trois indices : [3]

Au passage, une bonne habitude à prendre en perspective d'un passage à la vraie 3D – l'explosion de « pixels » est en 2D – consiste à donner les points d'une surface dans l'ordre qui évite l'élimination de la surface par culling des faces arrières. Autrement dit, donner les points dans le sens trigonométrique. Pour vérifier que ce sens est respecté, il est possible d'activer le culling, par défaut désactivé :

```
gl.enable(gl.CULL_FACE);
gl.cullFace (gl.BACK);
```

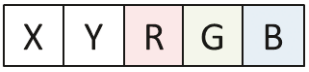
Le VS est alimenté en données relatives au point courant via des attributs. Ces données sont extraites de ARRAY_BUFFER, un buffer interne de WebGL auquel bindBuffer () permet d'associer un buffer créé avec createBuffer () et alimenté en données avec bufferData () (ou partiellement alimenté avec bufferSubData (), comme il en sera question dans le second article) :

```
var vertices = [
  -0.5, 0.5,  1.0,  0.0,  0.0,
  0.5,  0.5,  1.0,  0.0,  0.0,
  0.5,  -0.5, 1.0,  0.0,  0.0,
  -0.5, -0.5, 1.0,  0.0,  0.0,
];
var bufferV;

bufferV = gl.createBuffer ();
gl.bindBuffer (gl.ARRAY_BUFFER, bufferV);
gl.bufferData (gl.ARRAY_BUFFER, new Float32Array (vertices), gl.STATIC_DRAW);
```

Choix a été fait d'utiliser des indices en plus des points. Comme les points, les indices doivent être fournis sous la forme d'un tableau qui sert à alimenter un autre buffer, pour sa part associé au buffer interne de WebGL ELEMENT_ARRAY_BUFFER :

```
var indices = [
  0, 3, 1,
  1, 3, 2
];
var bufferI;

bufferI = gl.createBuffer ();
gl.bindBuffer (gl.ELEMENT_ARRAY_BUFFER, bufferI);
gl.bufferData (gl.ELEMENT_ARRAY_BUFFER, new Uint16Array (indices), gl.STATIC_DRAW);
```

Retour aux points. Un point se présente sous la forme d'une série de nombres décimaux respectant un format de point dont la définition implicite est libre. En l'espèce, le format de point est le suivant :



Toutefois, l'abscisse, l'ordonnée et la couleur d'un point ne sont pas les seules données dont le VS a besoin pour effectuer le travail attendu de lui. En effet, on souhaite qu'il transforme ce point en lui appliquant une rotation, puis un zoom, puis une translation. La particularité est que ces transformations sont identiques pour les quatre points des deux TRIANGLES composant un « pixel ». Comment le VS peut-il accéder à l'angle β, au facteur de zoom Z et aux composantes du vecteur de translation dX et dY propre au « pixel » courant ? Et d'abord, comment pourrait-il deviner à quel « pixel » le point courant appartient ?
La réponse est simple : le VS ne peut pas savoir à quel « pixel » le point courant appartient. Il travaille au niveau du point, pas de la primitive, et encore moins de l'objet dont cette dernière relève dans le programme principal – le fameux « pixel ». Par conséquent, dans un contexte où le VS va être appelé en rafale pour transformer tous les points de tous les « pixels » en une fois, si donnée supplémentaire, il faut lui transmettre via :

- Une uniforme si jamais cette donnée est la même pour tous les points qu'il doit traiter ;
- Un attribut – donc via ARRAY_BUFFER – si jamais cette donnée est propre au point courant.

Bref, β, Z, dX et dY doivent être rajoutées dans ARRAY_BUFFER.

L'organisation des données

Deux schémas peuvent être envisagés pour intégrer , Z, dX et dY à ARRAY_BUFFER. Dans le premier schéma (à gauche), ces données sont rajoutées après les données des points, alors que dans le second (à droite), elles sont rajoutées après les données de chaque point : [4]
Le choix du schéma peut constituer un enjeu pour les performances. Du fait qu'il vaut mieux regrouper les données qui n'ont pas besoin d'être modifiées durant l'animation, d'une part, et les données qui ont besoin d'être modifiées durant l'animation, d'autre part, on opte ici pour le premier schéma – le second article de cette série reviendra là-dessus.
Le contenu de ARRAY_BUFFER ainsi organisé, il faut porter cette organisation à la connaissance du VS. Comme le format de point ; cela se fait de manière implicite en définissant des attributs qui sont alimentés avec les bonnes données :

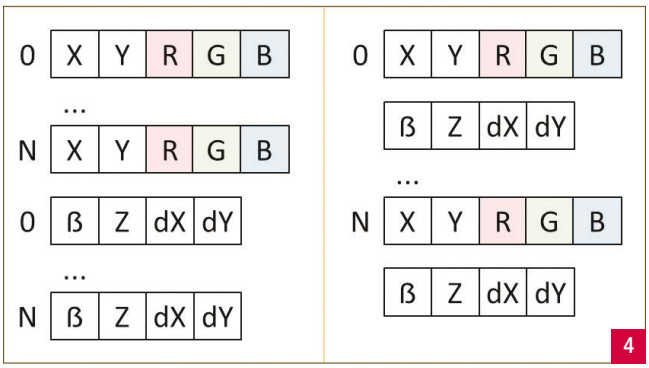
- A_xy de type vec2 pour X et Y (accessibles via A_xy.x et A_xy.y dans le code du VS, respectivement) ;
- A_rgb de type vec3 pour R, G et B (accessibles via A_rgb.x, A_rgb.y et A_rgb.z dans le code du VS, respectivement) ;
- A_bzdxdy de type vec4 pour , Z, dX et dY (accessibles via A_bzdxdy.x, A_bzdxdy.y, A_bzdxdy.z, A_bzdxdy.w dans le code du VS, respectivement).

Ces attributs doivent être déclarés dans le code du VS et associés à ARRAY_BUFFER dans le programme principal. Dans le code du VS, ils sont simplement déclarés avant main () :

```
attribute vec2 A_xy;
attribute vec3 A_rgb;
attribute vec4 A_bzdxdy;
```

Dans le programme principal, une fois les shaders compilés et rattachés à un programme compilé, ils sont associés à des variables par getAttribLocation () (une variable du programme principal ne doit pas nécessairement prendre le nom de la variable du shader à laquelle elle est associée, mais c'est plus pratique pour s'y retrouver) :

```
var A_xy, A_rgb, A_bzdxdy;

A_xy = gl.getAttribLocation (program, "A_xy");
```



```
A_rgb = gl.getAttribLocation (program, "A_rgb");
A_bzdxdy = gl.getAttribLocation (program, "A_bzdxdy");
```

Ces variables sont ensuite associées à ARRAY_BUFFER via vertexAttribPointer 0, mais il en sera question dans le second article de cette série. Pour alimenter le VS, il ne reste plus qu'à lui transmettre une matrice de transformation générale qui s'applique à tous les « pixels ». Cette matrice sert pour ajuster l'aspect ratio, c'est-à-dire pour s'assurer que la projection d'un pixel carré dans l'espace sera bien carrée à l'écran. Quand bien même la surface du canvas que WebGL considère correspondre à une surface de 2.0 x 2.0 dans l'espace ne soit pas carrée. Par exemple, à l'occasion d'un rendu dans un canvas de 800×400, le rendu sans ajustement de l'aspect ratio en haut et avec ajustement de l'aspect ratio en bas : [5]. Il convient donc de corriger les coordonnées des points d'un facteur qui dépend de la situation pour restituer l'espace à l'écran sans déformation : [6]
La matrice permet aussi d'appliquer une transformation identique à tous les « pixels », en l'occurrence leur appliquer une rotation autour du centre de l'écran. Une fonction getWorldMatrix 0 renvoie cette matrice :

```
function getWorldMatrix (angle, zoom, width, height) {
  var mWorld;

  zoom = 1.0;
  width = CANVAS_WIDTH;
  height = CANVAS_HEIGHT;
  mWorld = [
     Math.cos (angle), Math.sin (angle), 0.0, 0.0,
     - Math.sin (angle), Math.cos (angle), 0.0, 0.0,
     0.0, 0.0, 1.0, 0.0,
     0.0, 0.0, 0.0, 1.0
  ];
  if (width > height) {
     mWorld[0] *= height * zoom / width;
     mWorld[1] *= zoom;
     mWorld[4] *= height * zoom / width;
     mWorld[5] *= zoom;
  }
  else {
     mWorld[0] *= zoom;
     mWorld[1] *= width * zoom / height;
     mWorld[4] *= zoom;
     mWorld[5] *= width * zoom / height;
  }
  return (mWorld);
}
```

S'agissant d'une donnée identique pour tous les points transformés, la matrice peut être transmise au VS via une uniforme, en l'occurence U_mW. Comme les attributs, cette uniforme est déclarée avant main 0 dans le VS :

```
uniform mat4 U_mW;
```

Dans le programme principal, U_mW est accédée par l'intermédiaire d'une variable associée par getUniformLocation 0 :

```
U_mW = gl.getUniformLocation (program, "U_mW");
```

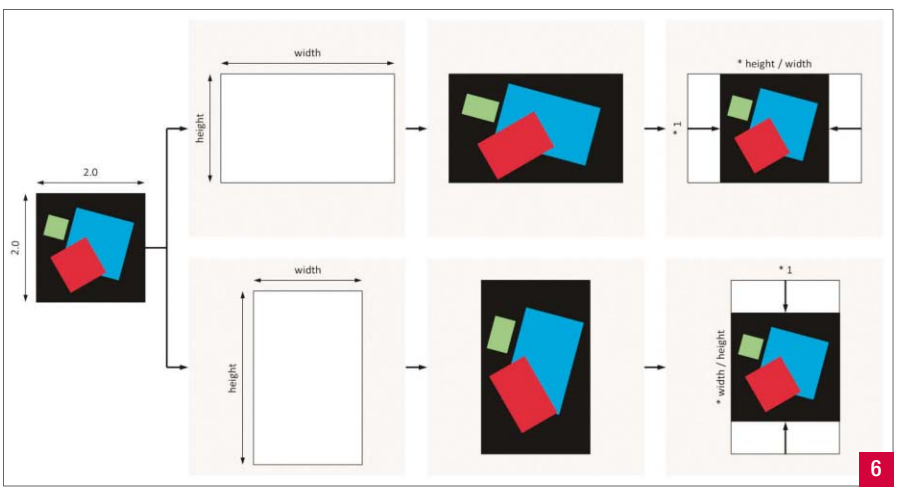
De la même manière, une uniforme U_a correspondant au degré de transparence, l'alpha, est définie dans le VS, et associée à une variable U_a dans le programme principal. En effet, l'alpha est le même pour tous les « pixels » qui disparaîtront de l'écran à la même vitesse tandis que cet alpha passera de 1.0 (opacité totale) à 0.0 (transparence totale).

Le code du VS

Mis à plat, le code du VS est le suivant :

```
attribute vec2 A_xy;
attribute vec3 A_rgb;
attribute vec4 A_bzdxdy;
uniform mat4 U_mW;
varying lowp vec3 V_rgb;

void main (void) {
  mat4 m;

  m = mat4 (1.0);
  m[0][0] = cos (A_bzdxdy.x) * A_bzdxdy.y;
  m[1][0] = - sin (A_bzdxdy.x) * A_bzdxdy.y;
  m[0][1] = sin (A_bzdxdy.x) * A_bzdxdy.y;
  m[1][1] = cos (A_bzdxdy.x) * A_bzdxdy.y;
```


5



6

```
  m[3][0] = A_bzdxdy.z;
  m[3][1] = A_bzdxdy.w;
  gl_Position = U_mW * m * vec4 (A_xy.xy, 0.0, 1.0);
  V_rgb = A_rgb;
}
```

Ce code est très simple, car il n'effectue qu'une multiplication de matrices : une matrice de transformation m propre au « pixel » – donc au point – est multipliée par la matrice des coordonnées du point, puis la matrice de transformation générale U_mW est multipliée au résultat. Autrement dit, le point est d'abord transformé localement puis globalement, comme il se doit.

A ce point, il convient de faire un aparté sur les matrices. L'usage des matrices dans un shader est particulièrement contre-intuitif pour qui a appris à en adresser les éléments en ligne puis en colonne. En effet, dans une variable de type matrice d'un shader, les éléments s'adressent en colonne puis en ligne :

```
mat4 m;

m[2][3] = 100.0; // L'élément figurant à la 3ème colonne ([2]) de la 4ème ligne ([3]) passe à 100.0
```

Bref, il faut se représenter l'opération consistant à multiplier m par vec4 (A_xy.xy, 0.0, 1.0) ainsi (les éléments étant numérotés en donnant la colonne puis la ligne) : [7]

Dans le code du shader, la multiplication de la matrice mat4 m par un vecteur vec4 v s'écrit m * v, et non l'inverse. Attention ! Dans le programme principal en JavaScript, l'alimentation d'une uniforme de type mat4 par un tableau de valeurs se fait en colonne :

```
var m = [
  m00, m01, m02, m03,
  m10, m11, m12, m13,
  m20, m21, m22, m23,
  m30, m31, m32, m33
];
var U_m = gl.getUniformLocation (program, "U_m");
gl.uniformMatrix4fv (U_m, false, m);
```

Noter que l'argument de transposition doit toujours être false, car la transposition ne marche pas.

Retour à la transformation d'un point d'un « pixel ». Nous avons déjà vu comment appliquer une rotation à un point autour d'un centre. Toutefois, c'était dans un repère dont l'axe des abscisses est orienté de gauche à droite et l'axe des ordonnées est orienté de haut en bas. Dans le contexte de WebGL, l'axe des abscisses est pareillement orienté, mais l'axe des ordonnées est orienté à l'inverse, de bas en haut. Par conséquent, appliquer à un point A une rotation d'angle autour d'un centre O produit un point B tel que :

- xB = xO + (xA – xO) * cos (β) – (yA – yO) * sin (β)
- yB = yO + (yA – yO) * sin (β) + (yA – yO) * cos (β)

La rotation doit être combinée au zoom, les deux se déroulant simultanément. Enfin, la translation peut être effectuée. C'est l'ordre dans lequel les transformations se déroulent quand elles sont décrites par une matrice multipliée à celle qui contient les coordonnées du point :

m[0][0]	Z * cos (β)	m[1][0]	- Z * sin (β)	m[2][0]	0	m[3][0]	dX
m[0][1]	Z * sin (β)	m[1][1]	Z * cos (β)	m[2][1]	0	m[3][1]	dY
m[0][2]	0	m[1][2]	0	m[2][2]	1	m[3][2]	0
m[0][3]	0	m[1][3]	0	m[2][3]	0	m[3][3]	1

Cette matrice est assemblée dans le code du VS, récupérant β, Z, dX et dY dans l'attribut via lequel elles parviennent :

β	A_bzdxdy.x
Z	A_bzdxdy.y
dX	A_bzdxdy.z
dY	A_bzdxdy.w

Au terme de ses calculs, le VS doit livrer deux données qui intéressent le FS. En effet, le FS doit déterminer les pixels que la projection du TRIANGLES occupe à l'écran, puis les colorier en interpolant les couleurs des sommets de ce TRIANGLES. Le VS doit donc livrer :

- Les coordonnées transformées du point. Pour cela, le VS doit alimenter la variable système gl_Position. S'agissant d'une variable système, elle n'a pas à être déclarée.
- La couleur du point. Pour cela, le VS doit alimenter une variante, une variable dont le type indique au FS qu'il doit en déterminer la valeur pour le pixel courant en interpolant les valeurs qu'elle prend aux sommets du TRIANGLES. Cette variable V_rgb est déclarée avant main 0, tant dans le code du VS que dans le code FS pour faire le lien.

Ce qui conduit à aborder le code du FS.

Le code du FS

Mis à plat, le code du FS est le suivant :

```
varying lowp vec3 V_rgb;
uniform lowp float U_a;

void main (void) {
  gl_FragColor = vec4 (V_rgb, U_a);
}
```

Ce code est totalement trivial, puisqu'il consiste à demander au FS d'afficher un pixel dont la couleur est celle résultant de l'interpolation déjà évoquée, sans plus. Pour cela, le FS doit alimenter une variable système gl_FragColor. L'initialisation est terminée. Il ne reste plus qu'à animer…

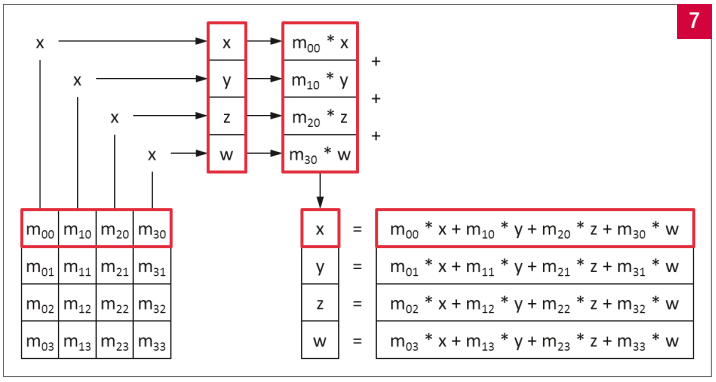
L'exemple

Rendez-vous à l'URL suivante pour accéder à une page de test minimaliste (un simple « pixel » est affiché après l'initialisation décrite à l'instant). Vous pourrez visualiser le code et le récupérer pour travailler avec.

http://www.stashofcode.fr/code/shader-explosion-de-pixels-avec-webgl-1/test.html •

Suite le mois prochain

7



Débuter avec **Laravel**

• Mathilde Luce-Lucas
Etudiante WebDesign
3ème année ICAN
• Morganne Lecordonnier
Etudiante WebDesign
3ème année ICAN

Laravel est un framework PHP fortement inspiré par Symfony. Dans cet article, vous découvrirez comment installer et mettre en place votre premier projet Laravel.

Installer Composer

Afin d'installer Laravel, vous aurez besoin de télécharger Composer : https://getcomposer.org/

Composer est un outil de gestion de dépendances PHP. En soit, il vous permet de déclarer les *library* dont votre projet a besoin, et le programme s'occupera de les installer et les mettre à jour. Une fois téléchargé, déplacez le fichier composer.phar sur votre serveur virtuel, soit htdocs si vous utilisez MAMP/WAMP ou XAMPP.

Ouvrez Terminal sur Macintosh, ou Command Prompt sur Windows.

Accédez à votre serveur virtuel en entrant le chemin dans votre fenêtre de commande ; par exemple, sur Mac avec MAMP :

```
cd/Applications/MAMP/htdocs
```

Ou avec Xamp, sur Windows :

```
cd C:/xampp/htdocs
```

À adapter à votre environnement de programmation. Pour installer Composer globalement, utilisez cette commande :

```
sudo mv composer.phar /usr/local/bin/composer
```

Créer un nouveau projet

Pour créer un nouveau projet, assurez-vous de bien être dans votre dossier htdocs. Utilisez cette commande pour lancer le projet :

```
composer create-project laravel/laravel <name>
```

<name> est le nom de votre projet. Vous pouvez choisir n'importe quel nom, mais pour les besoins du tutoriel, ici nous l'appellerons StarterLaravel. L'installation peut prendre un certain temps.

Après l'installation, si vous visitez localhost/StarterLaravel/public sur votre navigateur, vous pourrez normalement voir la page par défaut de Laravel. Avant de continuer, il est important de se familiariser avec l'architecture de Laravel. Nous avons ici affaire à un framework avec une structure MVC :

- app/ est la base des codes de l'application, son contenu. Il contient notamment ses controllers ;
- bootstrap/ est, comme son nom l'indique, un répertoire de commandes qui introduit les premières instructions leur permettant d'accéder au reste de l'application ;
- config/ contient toute les données permettant de paramétrer l'application ;
- database/ contient les fichiers de migration (sur lesquels nous reviendrons plus tard) ;
- public/ contient les assets de l'application (images, JS, CSS). C'est aussi lui qui fait le render des views ;
- resources/ contient les views sans compilation, les pré-processeurs, etc (SASS, Blade, Jade, CoffeeScript…) ;
- vendor/ contient les fichiers de dépendances Composer, comme expliqué plus tôt ;
- storage/ contient le cache de l'application, les logs de sessions etc.

Dans ce tutoriel, nous agirons sur des fichiers d'abord à la racine du projet, puis dans les répertoires app, config, database, et resources.

En vue de créer un nouveau formulaire, veuillez trouver le fichier composer.json à la racine de votre projet. Dans "require-dev", vous devrez rajouter la dépendance html form, comme ce qui suit :

```
"require-dev": {
    "fzaninotto/faker": "~1.4",
    "mockery/mockery": "0.9.*",
    "phpunit/phpunit": "~4.0",
    "symfony/css-selector": "2.8.*|3.0.*",
    "symfony/dom-crawler": "2.8.*|3.0.*",
    "laravelcollective/html": "5.2.*"
},
```

Dans votre fenêtre de commande, naviguez dans le dossier de votre projet :

```
cd /Applications/MAMP/htdocs/StarterLaravel
```

Puis exécutez cette commande afin d'installer la dépendance :

```
composer update
```

L'opération peut prendre un moment.

Dans config/app.php, trouvez ligne 124 " *'providers' => [* "

À la fin du tableau, ajoutez cette ligne :

```
Collective\Html\HtmlServiceProvider::class,
```

Puis trouvez ligne 172 " 'aliases' => ["

À la fin du tableau, ajoutez ces deux lignes :

```
'Form' => Collective\Html\FormFacade::class,
'Html' => Collective\Html\HtmlFacade::class,
```

Nous venons d'installer Html/Form car ces deux éléments ne sont malheureusement plus inclus dans Laravel depuis la sortie de sa version 5.0.

Formulaire

Les fichiers Blade sont rangés dans le dossier resources/views/.

Pour ce tutoriel et par souci d'efficacité, nous agirons directement dans le fichier welcome.blade.php. Evidemment, il est conseillé de créer vos propres fichiers de vues si vous souhaitez optimiser votre projet Laravel.

Premièrement, trouvez le fichier routes.php dans app/http/

Vous y trouverez ce code :

```
Route::get('/', function () {
    return view('welcome');
});
```

Ici, il est dit que la route (l'url) "/", soit la racine du site (communément appelée homepage), renvoie la vue "welcome" (faisant référence au fichier welcome.blade.php). Les données sont récupérées de la manière GET. Comme nous avons besoin d'un controller pour gérer nos actions, nous devons changer ce code en :

```
Route::get('/', 'StarterFormController@getInfos');
```

Ici, plutôt que de faire appel à une fonction unique, la homepage appellera la fonction getInfos() dans le controller 'StarterFormController' que nous allons créer de ce pas. Créez un nouveau fichier dans app/Http/Controller appelé StarterFormController.php.

```
namespace App\Http\Controllers;

use Illuminate\Foundation\Bus\DispatchesJobs;
use Illuminate\Routing\Controller as BaseController;
use Illuminate\Foundation\Validation\ValidatesRequests;
use Illuminate\Foundation\Auth\Access\AuthorizesRequests;
use Illuminate\Foundation\Auth\Access\AuthorizesResources;
```

Premièrement, on configure son namespace et le chargement de ce dont il a besoin :

```
class StarterFormController extends BaseController {
    use AuthorizesRequests, AuthorizesResources, DispatchesJobs, ValidatesRequests;
}
```

En dessous de la liste de "use", on ajoute la class StarterFormController :

```
class StarterFormController extends BaseController {
    use AuthorizesRequests, AuthorizesResources, DispatchesJobs, ValidatesRequests;
    public function getInfos(){
        return view('welcome');
    }
}
```

Enfin, ajoutez la fonction "getInfos()" qui se chargera d'afficher la page welcome.blade.php.

Blade form

Maintenant, rendez-vous dans les dossiers ressources/views/ et ouvrez welcome.blade.php. C'est dans ce fichier que nous créerons notre formulaire.
Admettons que l'on souhaite créer un formulaire où l'utilisateur renseignerait son nom, prénom, et émail. On utilisera le code suivant :

```
{!! Form::open(['url' => 'validation' ]) !!}
 {!! Form::label('gender', 'Genre :' ); !!}
 {!! Form::radio('gender', 'M'); !!} M.
 {!! Form::radio('gender', 'Mme', true); !!} Mme.
 {!! Form::radio('gender', 'Mlle'); !!} Mlle.
 <br />
 {!! Form::label('name', 'Nom :' ); !!}
 {!! Form::text('name', 'Luce' ); !!}<br/>
 {!! Form::label('surname', 'Prénom :' ); !!}
 {!! Form::text('surname', 'Mathilde' ); !!}<br/>
 {!! Form::label('email', 'Email :' ); !!}
 {!! Form::email('email', 'lolcats@lucelucas.work' ); !!}<br/>
```

Voyez cette ligne la ligne toute première : {!! Form::open(['url' => 'validation']) !!}
"Url" est l'action du formulaire. Ici, on dit que notre formulaire redirigera vers la page de validation, qui nous servira à l'affichage des informations saisies. Retrouvez le fichier app/Http/routes.php. Ajoutez cette ligne :

```
Route::post('/validation', 'StarterFormController@validateForm');
```

Comme expliqué précédemment, la route "validation" fera appel au même controller, mais appellera cette fois la fonction validateForm().
ValidateForm() s'occupera de la validation du formulaire, mais également de l'appel au Model pour faire rentrer ses informations en base de donnée.
Nous allons désormais créer un fichier pour gérer les requêtes du formulaire. Dans votre command prompt, entrez la ligne de code suivante :

```
php artisan make:request StarterFormRequest
```

Dans app/Http/Requests/, un nouveau fichier appelé StarterFormRequest.php a normalement été créé. Il se chargera du traitement et de la validité des données. Premièrement, trouvez la fonction authorize() (ligne 14), et changez son contenu en return true, comme ci-suit :

```
public function authorize(){
  return true;
}
```

C'est dans la fonction rules() que l'on définira l'ensemble de nos restrictions quant au formulaire. Par exemple :

```
return [
  'name' => 'required | max:20',
  'email' => 'required',
];
```

Name sera donc limité à 20 caractères, et les champs nom et émail seront tous les deux requis.
Retrouvez toute la liste des lois de validation sur la documentation officielle. https://laravel.com/docs/5.1/validation#available-validation-rules
Vous pouvez désormais tester votre formulaire ; si vous ne respectez pas vos contraintes, le controller vous renverra automatiquement sur la page de formulaire.
Nous pouvons désormais créer la page de validation.
Dans resources/views/ créez un nouveau fichier : validation.blade.php
Vous pouvez le laisser vide pour le moment.
Dans votre fichier StarterFormController.php, situé dans app/http/Controllers/, ajoutez la fonction en charge de l'affichage de la view :

```
public function validateForm(StarterFormRequest $request){
  return view('validation');
}
```

Ajoutez, avant le début de la classe, cette ligne :

```
use App\Http\Requests\StarterFormRequest;
```

Vous pouvez tester votre formulaire : s'il est valide, vous serez redirigé vers une page blanche (validation.blade.php).

Afficher les données

Pour passer les données envoyées par le formulaire à la vue, il faut les récupérer grâce au controller. Pour cela, on greffera un tableau au retour de la vue dans la fonction validateForm, comme il suit :

```
public function validateForm(StarterFormRequest $request){
  return view('validation', [
```

```
    "gender" => $request->input('gender'),
    "name" => $request->input('name'),
    "surname" => $request->input('surname'),
    "email" => $request->input('email')
  ]
  );
}
```

Ici, on déclare des variables et on leur donne une valeur ; ce qui équivaudrait à ceci en PHP classique :

```
$gender = $request->input('email');
```

Pour afficher ces données dans la vue, il suffit d'utiliser les variables Blade correspondant aux noms qu'on leur a donné :

```
<p>Genre : {!! $gender !!}</p>
<p>Nom : {!! $name !!}</p>
<p>Prénom : {!! $surname !!}</p>
<p>Email : {!! $email !!}</p>
```

Base de donnée et présentation d'Eloquent

Eloquent est l'ORM de Laravel. Un ORM (Object-Relational Mapping) est une technique qui permet de manipuler du data provenant de bases de données sans faire de requêtes, simplement en manipulant un objet. Eloquent utilise des "Models" pour gérer les tables de sa base de données.
Puisqu'Eloquent ne s'occupe pas de la création et/ou configuration des bases de données, il va nous falloir créer la nôtre nous-même.
Pour continuer, rendez-vous sur votre interface PhpMyAdmin et créez une nouvelle base de donnée. Pour ce tutorial, nous la nommerons : EloquentForm
Pour établir la connexion à la base de donnée, veuillez trouver le fichier Database.php dans le dossier config/ de votre projet. Vous aurez probablement à modifier les lignes 59 à 61 du fichier avec vos informations personnelles, comme ci-dessous :

```
'database' => env('DB_DATABASE', 'EloquentForm'),
'username' => env('DB_USERNAME', 'root'),
'password' => env('DB_PASSWORD', 'root'),
```

Sur Windows, la connexion à la base de données n'a pas besoin de mot de passe. Écrivez plutôt :

```
'password' => env('DB_PASSWORD', ''),
```

À la racine de votre projet, vous trouverez également un fichier .env qu'il faut éditer comme suit :

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=EloquentForm
DB_USERNAME=root
DB_PASSWORD=root
```

Laravel et Eloquent utilisent un système de "migrations" pour versionner la base de données. Ces migrations permettent également, pour les projets collaboratifs, de mettre à jour les bases de données sur les machines de chacun des développeurs facilement. Les migrations sont entrées à la fois en base de données, et sous forme de schéma dans des fichiers PHP que vous pourrez trouver dans database/migrations/
Pour créer la table de migration en base de données, utilisez votre interface de ligne de commande :

```
php artisan migrate:install
```

Mise en place de la base de données :

```
php artisan make:migration formulaire_data
```

Retrouvez le fichier généré dans database/migrations/[timestamp]_formulaire_data.php.
Trouvez la fonction up() ligne 13 du fichier et éditez-la comme il suit :

```
public function up() {
  Schema::create('tableFormulaire', function(Blueprint $table){
    $table->increments('id');
    $table->string('gender', 255);
    $table->string('name', 255);
    $table->string('surname', 255);
    $table->string('email', 255)->unique();
  });
}
```

Nous venons ainsi de créer les champs de notre base de données. Pour retrouver tous les champs possibles et leurs syntaxes, allez sur la documentation officielle de Laravel à cette adresse : https://laravel.com/docs/master/migrations#creating-columns .
Revenir en arrière nous ferait supprimer la base de données fraîchement créée. La fonction down() sera donc comme ce qui suit :

```
public function down() {
  Schema::drop('tableFormulaire');
}
```

Pensez à supprimer les fichiers par default 2014_10_12_000000_create_users_table.php et 2014_10_12_100000_create_password_resets_table.php du directory, car nous n'en avons pas besoin.
Pour mettre à jour la base de données, tapez cette ligne dans votre command prompt :

```
php artisan migrate
```

À titre d'information, pour revenir en arrière, la commande serait :

```
php artisan migrate:rollback
```

Générez un model pour les données du formulaire. Utilisez cette ligne de commande :

```
php artisan make:model StarterForm
```

Dans app/ vous retrouverez un fichier StarterForm.php. Par défaut, Eloquent ajoutera un timestamp sur la data à son entrée dans la base de données. Pour bloquer cette information, ajoutez la ligne public $timestamps = false; dans le Model, comme ci-suit :

```
class StarterForm extends Model
{
    public $timestamps = false;
}
```

Ajoutez la ligne "use DB;" à la suite de "use Illuminate\Database\Eloquent\Model;"
Pour rentrer les données envoyées par le formulaire dans la base de don-

nées, nous utiliserons la fonction addValues() qui aura pour paramètre les valeurs envoyées du controller :

```
public function addValues($gender, $name, $surname, $email){
 DB::insert( 'insert into tableFormulaire (gender, name, surname, email) values (?, ?, ?, ?)',
   array(
     $gender,
     $name,
     $surname,
     $email
   )
 );
}
```

Si vous vous êtes perdus, voici la totalité du fichier app/StarterForm.php :

```
<?php

namespace App;
use Illuminate\Database\Eloquent\Model;
use DB;
class StarterForm extends Model{
 public $timestamps = false;
 public function addValues($gender, $name, $surname, $email){
  DB::insert( 'insert into tableFormulaire (gender, name, surname, email) values (?, ?, ?, ?)',
    array(
      $gender,
      $name,
      $surname,
      $email
    )
  );
 }
}
```

Retrouvez votre fichier StarterFormController.php dans app/Http/Controllers/.
Ajoutez la ligne "use App\StarterForm;" en dessous "use App\Http\Requests\StarterFormRequest;"
Ajoutez la ligne "use App\StarterForm;" en dessous de "use App\Http\Requests\StarterFormRequest;"
Directement dans le Controller, ajoutez la variable protégée d'instance :

```
protected $instance;
```

Puis dans la fonction validateForm(), ajoutez une condition pour vérifier l'instance de la class par $instance :

```
if($this->instance === null){
 $instance = new StarterForm;
 $this->instance = $instance;
}
```

Par la suite, nous faisons la liaison entre le controller et le model grâce à cette instance :

```
$instance->addValues($request->input('gender'), $request->input('name'),
$request->input('surname'), $request->input('email'));
```

Si vous vous êtes perdus, voici la totalité du fichier app/http/Controllers/StarterFormController.php :

```
<?php
namespace App\Http\Controllers;

use Illuminate\Foundation\Bus\DispatchesJobs;
use Illuminate\Routing\Controller as BaseController;
use Illuminate\Foundation\Validation\ValidatesRequests;
use Illuminate\Foundation\Auth\Access\AuthorizesRequests;
use Illuminate\Foundation\Auth\Access\AuthorizesResources;
use App\Http\Requests\StarterFormRequest;
use App\StarterForm;

class StarterFormController extends BaseController {
 use AuthorizesRequests, AuthorizesResources, DispatchesJobs, ValidatesRequests;
 protected $instance;

 public function getInfos(){
  return view('welcome');
 }
 public function validateForm(StarterFormRequest $request){
  if($this->instance === null){
   $instance = new StarterForm;
   $this->instance = $instance;
  }

  $instance->addValues($request->input('gender'), $request->input('name'),
$request->input('surname'), $request->input('email'));

  return view('validation', [
   "gender" => $request->input('gender'),
   "name" => $request->input('name'),
   "surname" => $request->input('surname'),
   "email" => $request->input('email')
  ]);
 }
}
```

Créer un **bot Facebook Messenger** en 15 minutes

Adrien ROCHEDY
Ingénieur d'étude .Net, **Nouveal e-santé**
Twitter : @adr_roch

nouveal e-santé

Les bots débarquent en masse dans tous les services de messagerie. Ceux-ci vont permettre de répondre automatiquement aux questions et demandes des utilisateurs qui vont communiquer en écrivant des messages directement dans la messagerie. L'utilisateur va pouvoir être aidé par un assistant virtuel lorsqu'il souhaitera commander des billets d'avion, des pizzas, connaître la météo et voir si des Pokémons rares se trouvent à côté de lui. Intéressons-nous au dernier sorti : les bots pour Facebook Messenger.

Un bot Messenger est simplement un serveur identifié par Facebook qui va recevoir des messages et choisir quelle réponse renvoyer. Pour en créer un c'est simple : ça se passe en 3 étapes. Créer le serveur de réponse, paramétrer la page Facebook et faire parler le bot. Nous allons voir comment créer un bot Messenger avec node.js et Heroku comme hébergeur.

Créer le serveur de réponse

Grâce à Heroku (racheté par Facebook) il est possible d'héberger dans le Cloud un bot Facebook en Node.js rapidement.

- Installer heroku toolbelt (https://toolbelt.heroku.com) sur votre ordinateur pour pouvoir lancer, stopper et paramétrer votre serveur. Vous avez besoin d'un compte Heroku (gratuit : https://www.heroku.com)
- Installer Node (https://nodejs.org)
- Vérifier la version de node dans la console

```
sudo npm install npm -g
```

- Créer un dossier contenant notre code pour le bot et créer une appli node (faire « entrer » pour accepter les paramètres par défaut).

```
npm init
```

- Installer les dépendances : Express, request (pour envoyer des messages) et body-parser (pour lire les messages).

```
npm install express request body-parser --save
```

- Créer un fichier index.js et écrire le code suivant.

```
var express = require('express') ;
var bodyParser = require('body-parser') ;
var request = require('request') ;
var app = express() ;

app.set('port', (process.env.PORT || 5000)) ;

// Process application/x-www-form-urlencoded
app.use(bodyParser.urlencoded({extended: false})) ;

// Process application/json
app.use(bodyParser.json()) ;

// Index route
app.get('/', function (req, res) {
  res.send('Hello world, je suis un chat bot') ;
}) ;

// for Facebook verification
app.get('/webhook/', function (req, res) {
  if (req.query['hub.verify_token'] === 'password_facile_a_retenir') {
    res.send(req.query['hub.challenge']) ;
  }
  res.send('erreur, mauvais token') ;
}) ;

// Spin up the server
app.listen(app.get('port'), function() {
  console.log('running on port', app.get('port')) ;
}) ;
```

La route « webhook » ci-dessus permet à Facebook d'authentifier votre bot et de l'associer à votre page.

- Créer un fichier Procfile et écrire la ligne ci-dessous. Grâce à ce fichier, Heroku saura par quelle page démarrer.

```
web: node index.js
```

- Dans le fichier package.json ajouter la ligne 1. "start": "node index.js", dans la partie scripts

```
"scripts": {
  "test": "echo \"Error: no test specified\" && exit 1",
  "start": "node index.js"
},
```

- Commiter et pusher le code avec Git. Heroku va automatiquement lancer le code dans le Cloud.

```
git init
git add .
git commit --m 'hello world'
heroku create
git push heroku master
```

Paramétrer l'application Facebook

- La première étape est de créer ou configurer une application Facebook sur le site https://developers.facebook.com/apps/ [1]

- Une fois l'application créée, ajoutez un produit à l'application et sélectionner « messenger ».
- Aller dans l'onglet « messenger » et la partie « Webhooks » cliquer sur le bouton « Setup Webhooks ».
- Vous allez insérer ici l'url de votre serveur Heroku. Vous définissez également le token qui va permettre à Facebook d'être sûr que votre serveur Heroku est bien le serveur destiné à recevoir et envoyer des messages pour votre page (dans le code d'index.js c'est le texte « password_facile_a_retenir »). Pensez à bien cocher toutes les permissions. [2]
- Récupérer le « Token d'accès de la Page ». (Si vous n'avez pas de page Facebook c'est le moment d'en créer une).
- Il faut ensuite envoyer une requête post à Facebook avec le token de votre page pour lui indiquer que votre page est prête à recevoir des messages. Vous pouvez le faire en ligne de commande ou vous pouvez aussi utiliser un outil de REST API comme postman pour passer cette commande post.

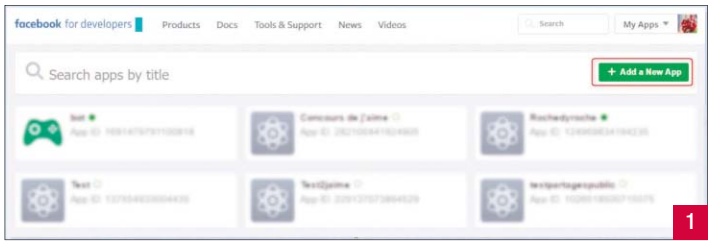
```
curl -X POST
"https://graph.facebook.com/v2.6/me/subscribed_apps?access_token=<PAGE_ACCESS_TOKEN>"
```

Faire parler le bot

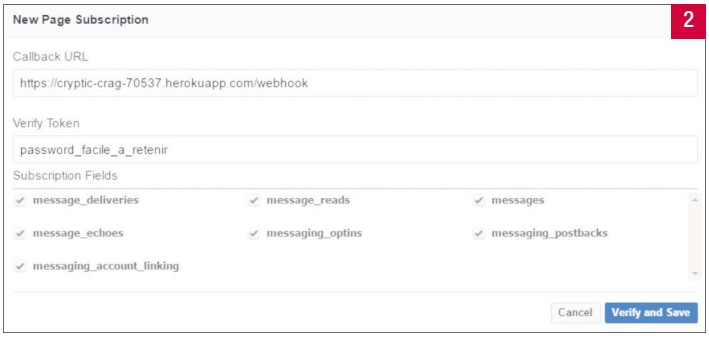
Maintenant que Facebook et Heroku peuvent communiquer entre eux on va pouvoir construire le bot.

- Ajoutez une entrée sur votre serveur (dans index.js) pour récupérer les messages. N'oubliez pas d'ajouter le token récupéré plus tôt.

```
var token =<PAGE_ACCESS_TOKEN>" ;
app.post('/webhook/', function (req, res) {
var messaging_events = req.body.entry[0].messaging ;
for (var i = 0; i < messaging_events.length; i++) {
  var event = req.body.entry[0].messaging[i] ;
  var sender = event.sender.id
  if (event.message && event.message.text) {
      var text = event.message.text ;
      sendTextMessage(sender, "Message reçu : " + text.substring(0, 200)) ;
      }
  }
  res.sendStatus(200) ;
})
```



Créer une nouvelle app facebook



- Créer la fonction permettant de renvoyer des messages.

```
function sendTextMessage(sender, text) {
  var messageText = { text:text } ;
  request({
    url: 'https://graph.facebook.com/v2.6/me/messages',
    qs: {access_token:token},
    method: 'POST',
    json: {
      recipient: {id:sender},
      message: messageText,
      }
    }, function(error, response, body) {
      if (error) {
        console.log('Une erreur est survenue : ', error) ;
      } else if (response.body.error) {
        console.log('Erreur: ', response.body.error) ;
      }
    }) ;
}
```

- Commiter/pusher

```
git add .
git commit -m 'mes premiers mots'
git push heroku master
```

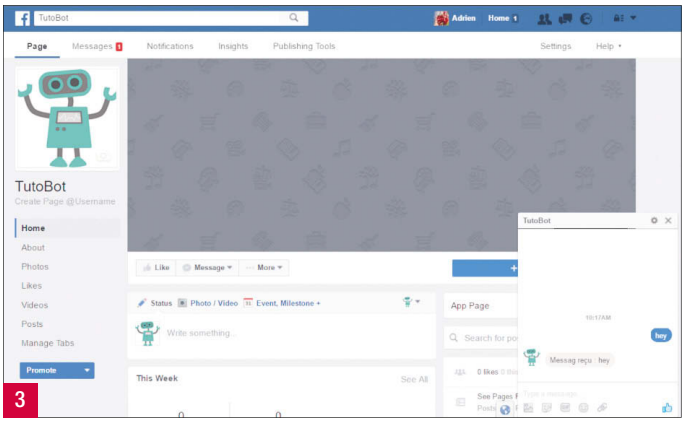
- Et allez sur votre page et commencer à chattez ;) [3]

Envoyer des messages structurés

Maintenant que votre bot nous répond, nous allons pouvoir envoyer des messages un peu plus intéressants. Facebook propose plusieurs types de messages qui peuvent être envoyés par le bot. Ces messages peuvent être utilisés pour proposer une liste de produits à un utilisateur ou le faire répondre à un questionnaire.

- Ajouter ce code dans index.js to permettant d'envoyer 2 « cartes ».

```
function sendCardMessage(sender) {
  var messageData = {
    "attachment": {
      "type": "template",
      "payload": {
        "template_type": "generic",
        "elements": [{
          "title": "Premiere carte",
          "subtitle": "Element 1 de la liste",
```



Chatter avec le bot

```
                "image_url": "http://messengerdemo.parseapp.com/img/rift.png",
                "buttons": [{
                  "type": "web_url",
                  "url": "https://www.messenger.com",
                  "title": "Visiter le site"
                }, {
                  "type": "postback",
                  "title": "Postback",
                  "payload": "Payload",
                }],
              }, {
                "title": "Deuxieme carte",
                "subtitle": "Element numero 2 de la liste",
                "image_url": "http://messengerdemo.parseapp.com/img/gearvr.png",
                "buttons": [{
                  "type": "postback",
                  "title": "Postback",
                  "payload": "Payload",
                }],
              }]
            }
          }
        }
        request({
          url: 'https://graph.facebook.com/v2.6/me/messages',
          qs: {access_token:token},
          method: 'POST',
          json: {
            recipient: {id:sender},
            message: messageData,
          }
        }, function(error, response, body) {
          if (error) {
            console.log('Erreur pendant l\'envoi du message ', error) ;
          } else if (response.body.error) {
            console.log('Erreur: ', response.body.error) ;
          }
        })
}
```

- Modifier le webhook pour récupérer le message permettant d'envoyer les 2 cartes.

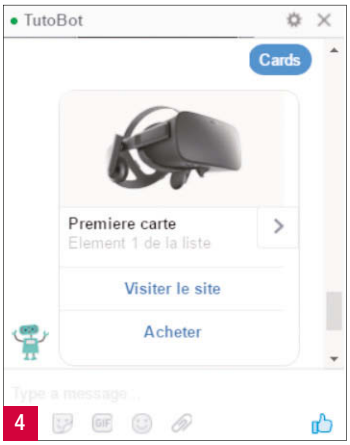
```
app.post('/webhook/', function (req, res) {
  var messaging_events = req.body.entry[0].messaging ;
  for (var i = 0; i < messaging_events.length; i++) {
    var event = req.body.entry[0].messaging[i] ;
    var sender = event.sender.id ;
    if (event.message && event.message.text) {
      var text = event.message.text ;
      if (text === 'Cards') {
        sendCardMessage(sender) ;
        Continue ;
      }
      sendTextMessage(sender, "Message reçu : " + text.substring(0, 200)) ;
    }
  }
  res.sendStatus(200) ;
})
```

Réagir au clic de l'utilisateur

Que se passe-t-il quand l'utilisateur clique sur un bouton ou une carte ? Modifions la webhook api une dernière fois pour renvoyer un message au postback des boutons.

```
app.post('/webhook/', function (req, res) {
var messaging_events = req.body.entry[0].messaging ;
for (var i = 0; i < messaging_events.length; i++) {
  var event = req.body.entry[0].messaging[i] ;
  var sender = event.sender.id ;
  if (event.message && event.message.text) {
    var text = event.message.text ;
      if (text === 'Cards') {
        sendCardMessage(sender) ;
        continue
      }
      sendTextMessage(sender, "Message reçu : " + text.substring(0, 200));
    }
    if (event.postback) {
      var text = JSON.stringify(event.postback) ;
      sendTextMessage(sender, "Postback reçu : "+text.substring(0, 200), token) ;
      continue ;
   }

  }


  res.sendStatus(200) ;
})
```

Git add, commit, et push. Maintenant quand vous enverrez 'Cards' à votre bot vous pourrez voir ça : [4]



Un message structuré

Aller plus loin

Voici donc comment créer rapidement un bot Messenger. Vous pouvez aller plus loin en le faisant valider par Facebook et ainsi le rendre accessible à tous. Il est aussi possible d'ajouter un bouton « Contactez-moi sur Messenger » sur votre site. L'avantage de créer un bot Facebook est de pouvoir utiliser la puissante « Graph API » de Facebook pour en savoir plus sur l'interlocuteur et ainsi lui proposer des réponses vraiment personnalisées. Enfin, la prochaine étape est d'ajouter du NLP (neuro-linguistic programming) pour que votre bot comprenne le langage naturel et puisse répondre à toutes les questions de vos utilisateurs. Facebook propose une api pour ça : wit.ai. •

Lien vers le repo git du code de l'article :
https://github.com/arochedy/bot-fb-messenger

Documentation officielle :
https://developers.facebook.com/docs/messenger-platform/quickstart
Faire valider son bot : https://developers.facebook.com/docs/messenger-platform/app-review
Créer un bouton « contactez-moi » :
https://developers.facebook.com/docs/messenger-platform/plugin-ref
Magazine online sur les bots : https://chatbotsmagazine.com/
Groupe Facebook de développeurs de bots : https://www.facebook.com/groups/chatbot/

Découvrez tangente le magazine des mathématiques

Pour mieux comprendre le monde

tangente
l'aventure mathématique

Aussi en version numérique !

Tous les deux mois chez votre marchand de journaux

tangente
l'aventure mathématique

Politique, économie, finance, jeux, musique, littérature, arts plastiques, architecture, **informatique**, physique, biologie, géographie : **les mathématiques sont partout !!!** Le magazine *Tangente* et ses hors séries vous aident à redécouvrir notre quotidien.

Les hors séries « kiosque »

4 fois par an, un hors série d'au moins 56 pages, explore un grand dossier de savoir ou de culture. **Derniers parus :** Les angles - Les graphes - Les démonstrations - Les fonctions - Maths des assurances - Maths et médecine - La Droite - Maths et Architecture - Les ensembles

Disponibles chez votre marchand de journaux ou avec **l'abonnement PLUS.**

Vous voulez vous rendre compte de ce qu'est la consultation numérique d'un numéro de *Tangente* ?

L'accès au numéro 167 vous est offert !

Rendez-vous sur http://tangente-mag.com (identifiez-vous)

La « Bibliothèque Tangente »

Pour les lecteurs les plus curieux, les articles des hors séries de *Tangente* sont repris et complétés dans la Bibliothèque Tangente, avec de magnifiques ouvrages d'environ 160 pages (prix unitaire 20 à 22 €), richement illustrés, disponibles

- sur la boutique du site **www.infinimath.com**
- chez votre libraire
- avec **l'abonnement SUPERPLUS.**

http://tangente-mag.com

DEMANDEZ UN ANCIEN NUMÉRO de *Tangente* - **Joignez juste 3 € de timbres**

À adresser à *TANGENTE* - 80 BD SAINT-MICHEL - 75006 PARIS avant le 31/03/17

NOM* . . . . . PRÉNOM* . . . . .

ADRESSE* . . . . .

CODE POSTAL* . . . . . VILLE* . . . . . PAYS . . . . .

MAIL* . . . . . TÉLÉPHONE* . . . . .

ABONNÉ À PROGRAMMEZ ❒ OUI ❒ NON PROFESSION . . . . .

Ecrire un agent conversationnel avec Microsoft Bot Framework

Eric Vernié
Fier d'être développeur

Le monde de l'informatique est en continuelle transformation, les bots, ou plus précisément les agents conversationnels qui seraient capables de répondre aux exigences du Test de Turing, en sont la parfaite illustration.

Même si les bots ne sont pas nouveaux, ils émergent au sein de la population des développeurs sous l'influence de deux facteurs prépondérants : le cloud et l'intelligence artificielle et sa démocratisation.

Mais s'il est facile d'utiliser le Cloud et de l'intelligence artificielle, il doit être également facile de développer un bot et de le déployer. C'est ce que propose le Microsoft Bot Framework ; fournir une plateforme, pour accélérer le développement et le déploiement de bots. Avec un portail pour le développeur, afin de le mettre à disposition sur les canaux de communications, tels que Skype, Slack, FaceBook Messenger, Telegram, Text/SMS, et d'autres services populaires.

Microsoft Bot Framework : https://dev.botframework.com/

Préparation de l'environnement

Le bot framework est basé sur des API REST décrites au format swagger. Néanmoins Microsoft fournit deux SDK de base, un pour .NET, et un pour node.js pour commencer à développer dans les meilleures conditions. Dans cet article, nous utiliserons le SDK .NET, ainsi que Visual Studio 2015 (à télécharger à cette adresse http://aka.ms/FreeVS2015). Le SDK .NET est disponible en open source sur github : https://github.com/Microsoft/BotBuilder. Sinon les binaires sont disponibles directement sous forme de package nugget

```
Install-package Microsoft.Bot.Builder
```

En revanche, pour démarrer le plus rapidement possible avec Visual Studio 2015, je vous conseille de télécharger un modèle de projet, disponible à cette adresse : http://aka.ms/bf-bc-vstemplate, et à installer tel quel dans ce répertoire.

```
USERPROFILE%\Documents\Visual Studio 2015\Templates\ProjectTemplates\Visual C#\
```

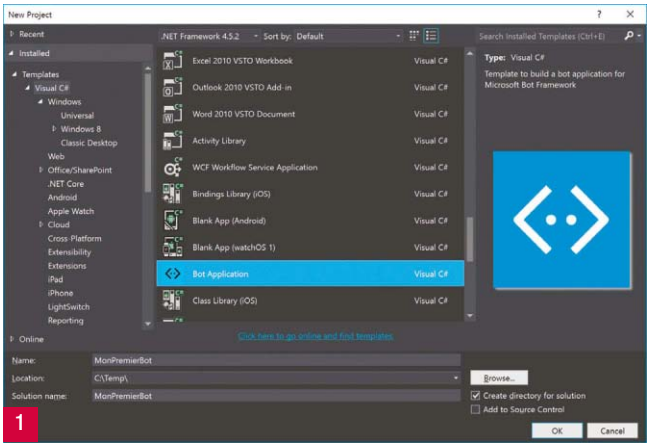
Enfin, pour tester le bot, installez un émulateur à partir de https://aka.ms/bf-bc-emulator

Notre premier bot

Dans Visual Studio 2015, vous avez donc accès à un nouveau modèle de projet comme illustré sur la figure suivante : [Fig.1]. Le type d'application utilisé pour notre Bot est de type traditionnel **Web Api ASP.NET** donc **RESTFul.** Ce projet étant déjà opérationnel il démarre un serveur Web associé (en l'occurrence IIS Express), et le bot à une adresse du type https://localhost:3979. Pour le tester, lancez l'outil **Microsoft Bot Channel Emulator**.

Toute conversation, débute par l'envoi d'une activité (**Activity**)

```
public async Task<HttpResponseMessage> Post([FromBody]Activity activity)

{
    if (activity.Type == ActivityTypes.Message)
    {
```



1

```
        ConnectorClient connector = new ConnectorClient(new Uri(activity.ServiceUrl));
        Activity reply = activity.CreateReply($"You sent {activity.Text} which was {length} characters");
        await connector.Conversations.ReplyToActivityAsync(reply);
    }
```

Cet objet **Activity,** porte un certain nombre de propriétés comme par exemple.

- **ChannelId :** "emulator", qui permet de déterminer quel est le canal qui invoque le Bot (Skype, FaceBook Messenger et autres) ;
- **From :** Permet d'identifier l'utilisateur du canal ;
- **Conversation :** Permet d'identifier la conversation en cours, ce qui peut être important afin de ne pas perdre le fil ;
- **ServiceUrl :** l'url du service ;
- **Text :** "Bonjour", le texte de l'utilisateur ;
- **Type :** "message". Le type message est le plus souvent celui qui est envoyé, mais il en existe d'autres surtout dans des scenarios qui impliquent plusieurs utilisateurs.

Le framework fournit un ensemble de fonctionnalités qui permettent de développer un Bot de manière plus naturelle à base de **Dialogues.**

En effet, le dialogue, ou plus précisément la classe **Dialog** est la pierre angulaire du processus conversationnel que l'on engage avec l'utilisateur, où l'échange de messages est le canal primaire pour l'interaction avec le monde extérieur. Il sera possible de chaîner des dialogues entre eux afin de créer une réelle conversation avec l'utilisateur. Les caractéristiques d'un objet Dialog, c'est qu'il maintient son état au travers de différentes instances de machines, permettant au Bot de monter à l'échelle en termes de performances si besoin est.

Commençons par un simple Dialog. Il suffit d'implémenter l'interface **IDialog** et de rendre la class **Serializable,** afin qu'elle puisse voyager sur différentes instances de machines.

```
[Serializable]
```

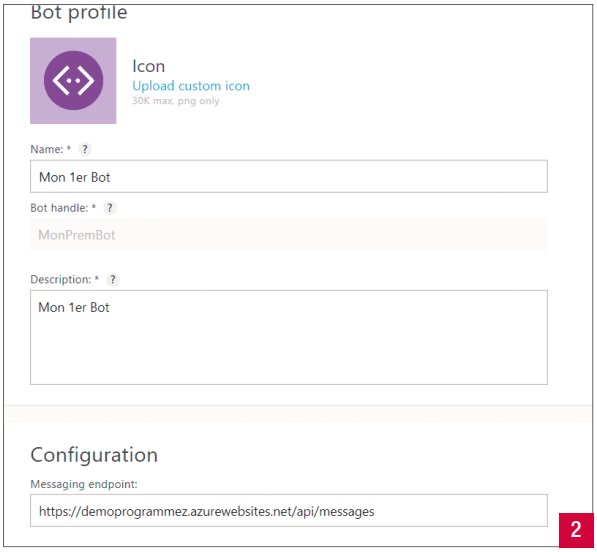
```
public class RootDialog : IDialog<object>
{
    public async Task StartAsync(IDialogContext context)
    {
        context.Wait(MessageReceivedAsync);
    }
    public virtual async Task MessageReceivedAsync(IDialogContext context, IAwaitable
<IMessageActivity> result)
    {
        var message = await result;
        await context.PostAsync($"vous avez dit {message.Text}");
        context.Wait(MessageReceivedAsync) ;
    }
```

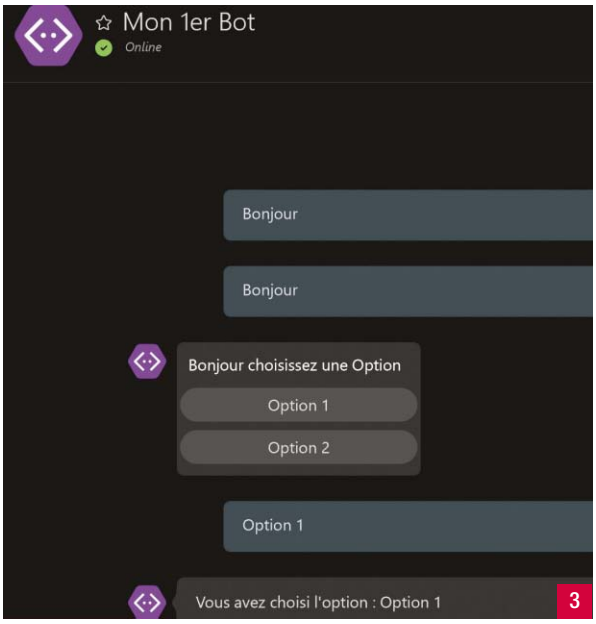
Au démarrage de la conversation (**StartAsync**), notre **RootDialog,** attend qu'un utilisateur engage une conversation (**var message = await result**). Notre bot lui répond (**PostAsync**) par l'intermédiaire d'un contexte que chaque Dialog possède. Ensuite on attend de nouveau une entrée de la part du même utilisateur **context.Wait(Message ReceivedAsync)**.
Finalement pour invoquer notre RootDialog, à partir de la conversation courante, on délègue l'activité à notre Dialog (**SendAsync**).

```
public async Task<HttpResponseMessage> Post([FromBody]Activity activity)
    {
        if (activity.Type == ActivityTypes.Message)
        {
            await Conversation.SendAsync(activity, () => new RootDialog());
        }
        else
        {
            HandleSystemMessage(activity);
        }
        return Request.CreateResponse(HttpStatusCode.OK);
    }
```

Il est possible et nécessaire de chaîner les dialogues entre eux. Dans ce second exemple, lorsque la RootDialog s'exécute, une Dialog à choix multiples s'affiche et en fonction du choix dans la méthode **ResumeAfter,** on appelle la Dialog appropriée (**context.call**)

```
public virtual async Task MessageReceivedAsync(IDialogContext context, IAwaitable<IMessage
Activity> result)
{
    var message = await result;
    PromptDialog.Choice(context, ResumeAfter, new List<string> { "Option 1 ", "Option 2" },
"Bonjour choisissez une Option");
    }
    private async Task ResumeAfter(IDialogContext context, IAwaitable<string> result)
    {
        var message = await result;
        await context.PostAsync($"Vous avez choisi l'option : {message}");
 if (message == "Option 1")
        {
            context.Call(new DialogOption1(), ResumeAfter);
        }
        else if(message == "Option 2")
        {
            context.Call(new DialogOption2(), ResumeAfter);
        }
```






```
        context.Wait(MessageReceivedAsync);
    }
```

Enregistrement de notre Bot dans le portail de développement

Après déploiement du bot sur le site Web https://demoprogrammez.azurewebsites.net/

Note : *Vous pouvez naturellement déployer votre Bot là où bon vous semble.*

Nous allons l'enregistrer sur le portail de développement.
Pour ce faire, il faut se connecter au site https://dev.botframework.com/bots/new
Remplir les champs nécessaires. [Fig.2]
Ne pas oublier de mettre le bon point de terminaison
https://demoprogrammez.azurewebsites.net/api/messages
Une fois enregistré, il faut choisir le ou les canaux de communication que vous souhaitez. Dans mon exemple, je l'ai branché sur Skype en l'ajoutant en tant que contact, je peux donc commencer à dialoguer avec lui. [Fig.3]. Développer, déployer un bot est désormais simple avec Microsoft Bot Framework. Dans un prochain article, nous verrons qu'il est également tout aussi simple de lui insuffler un peu d'intelligence. •

Créer une police d'icônes

• Yves Skrzypczyk
Professeur permanent à l'ESGI
Ecole Supérieure de Génie Informatique
2016 - 2017

Avant de pouvoir expliquer ce qu'est une police d'icônes il faut bien comprendre ce qu'est une police d'écriture. Dès lors nous pourrons comprendre tous les avantages de cette technique et de son mode de fonctionnement.

Une police d'écriture est en typographie un ensemble de glyphes, des représentations graphiques permettant la représentation d'un ensemble de caractères d'une langue. Une police d'écriture, communément appelée une **typo** permet donc de personnaliser une écriture à la fois sur un site WEB que sur un document word.
L'intérêt principal d'une typo, c'est sa **vectorisation**, sa capacité à changer de taille sans perdre en qualité :

Imaginez maintenant que l'on remplace un caractère par une icône, il nous sera alors possible de modifier facilement sa taille, sa couleur ou de rajouter des effets graphiques comme des ombres ou encore une couleur de fond :

Pour résumer, une police de font est une police dans laquelle nous avons remplacé les caractères (glyphes) par des icônes, pour bénéficier de tous leurs avantages :
- Optimisation du temps de chargement ;
- Facilité de mise en place et d'administration ;
- Compatibilité sur tous les navigateurs ;
- Vectoriel ;
- etc.

L'EXISTANT
Analyse de deux grands piliers

Font Awesome

Solution propulsée par le très connu framework **BOOTSTRAP** jusqu'à sa version 3 utilisant maintenant Glyphicons, une solution payante.

http://fontawesome.io/

Actuellement dans sa version 4.7.0, cette solution propose 675 icônes et fonctionne grâce à l'application d'une classe **fa** ainsi qu'une seconde classe avec le préfixe **fa-** , exemple :

```
<i class="fa fa-facebook"></i>
```

Fondation Icon Fonts

Solution propulsée par le très connu framework **FOUNDATION**.

http://zurb.com/playground/foundation-icon-fonts-3/

Actuellement dans sa version 3.0, cette solution propose 283 icônes et fonctionne grâce à l'application d'une classe avec le préfixe **fa-** , exemple :

```
<i class="fi-social-facebook"></i>
```

CRÉATION D'UNE ICÔNE
présentation de deux solutions

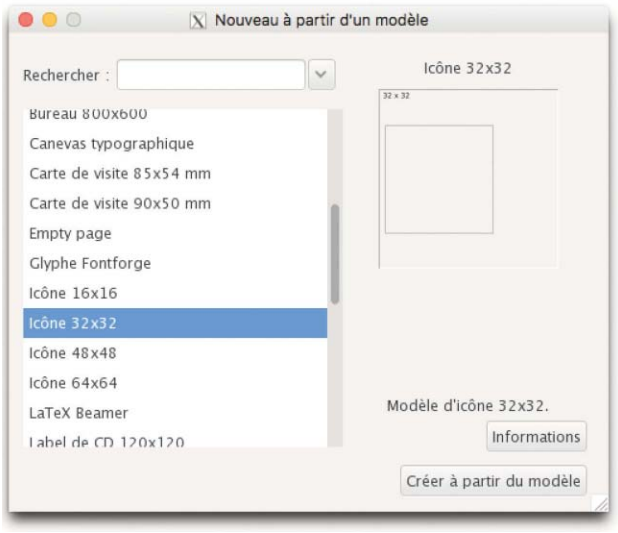
Outil : INKSCAPE

Pour créer nos propres icônes nous allons utiliser un logiciel gratuit équivalent à Illustrator. Inkscape est un logiciel de dessin vectoriel professionnel pour Windows, Mac OS X et GNU/Linux. Il est libre et gratuit

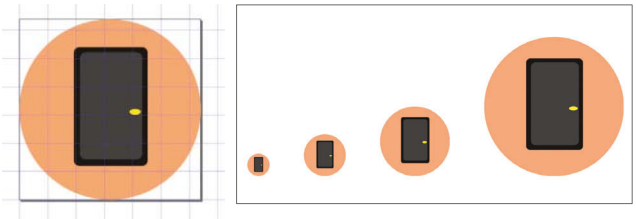
https://inkscape.org/fr/
Version : 0.91
OS utilisé : MAC SIERRA

Créer une image vectorielle

Ce document n'a pas pour objectif d'apprendre à réaliser une image vectorielle néanmoins voici le b.a.-ba:
Ouvrons inscape et créons un nouveau document à partir d'un modèle, l'icône 32x32



Affichons la grille afin de simplifier le positionnement **Affichage > grille**.
Il faut penser à enregistrer le document même vierge au format svg afin de ne pas avoir de mauvaises surprises et maintenant laissons libre cours à notre imagination :



Transformer un png en image vectorielle

Il est important de choisir une image dont les couleurs ne permettent pas une bonne compréhension de l'illustration car elles vont disparaitre pour avoir une icône monochrome, voici l'image que nous allons utiliser ici :

Après avoir ouvert l'image nous pouvons utiliser l'outil de vectorisation en allant dans **Chemin > Vectoriser le bitmap**
Nous aurons, à ce moment-là, le choix de nous baser sur la luminosité (composantes rouges, bleues et vertes), la détection du contour (détection des arêtes) ou la quantification des couleurs pour convertir l'image. Dans tous les cas il est intéressant de tester avec un aperçu afin de récupérer le svg qui convient le mieux à nos attentes.

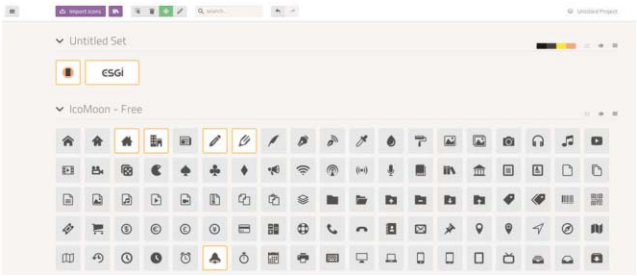
Une fois que le rendu nous convient, revenons sur l'interface principale afin d'enregistrer l'image au format vectoriel svg.

CRÉATION D'UNE POLICE D'ICÔNES
utilisation d'une application WEB

application : IcoMoon

https://icomoon.io/

Heureusement il existe des solutions Web permettant d'intégrer nos images vectorielles et de créer automatiquement une police afin de l'intégrer dans notre site Internet. Nous allons utiliser ici une solution gratuite **Icomoon.**
L'intérêt d'utiliser Icomoon, c'est que cette solution possède une quantité intéressante d'icônes prêtes à l'emploi. De plus l'application est intuitive et rapide.
Commençons par importer nos nouvelles icônes et sélectionnons ensuite les icônes que nous souhaitons ajouter à notre collection.



Une fois notre choix fait, nous pouvons télécharger la police dans plusieurs formats (des formats standards de police de caractères) en cliquant sur le bouton en bas à droite de l'application :

Le dossier contient un fichier CSS du nom de **style.css** ainsi qu'un dossier **fonts** contenant notre police d'icônes. Les autres éléments nous permettent d'avoir un aperçu rapide du résultat. Arrêtons nous rapidement sur le fichier CSS pour en comprendre le fonctionnement :

```
@font-face {
  font-family: 'icomoon';
  src:  url('fonts/icomoon.eot?8ph3ci');
  src:  url('fonts/icomoon.eot?8ph3ci#iefix') format('embedded-opentype'),
    url('fonts/icomoon.ttf?8ph3ci') format('truetype'),
    url('fonts/icomoon.woff?8ph3ci') format('woff'),
    url('fonts/icomoon.svg?8ph3ci#icomoon') format('svg');
  font-weight: normal;
  font-style: normal;
}
```

Nous avons ici le code permettant de mettre en place notre nouvelle police qui se trouve dans différents formats afin de nous assurer de sa compatibilité sur les différents navigateurs.

```
.icon-logo-esgi:before {
  content: "\e905";
}
```

Ensuite nous retrouvons la classe à appliquer en HTML afin d'afficher l'icône souhaitée. Cette ligne utilise la pseudo-classe :

A

before permettant d'insérer le contenu html \e905 dans notre balise HTML :

```
<span class="icon-logo-esgi"></span>
```

CONCLUSION

Nous pouvons nous rendre compte maintenant de la facilité de mise en place d'une solution permettant à la fois de bénéficier des avantages de la vectorisation mais aussi de la rapidité de développement qui s'en suit.

La police d'icônes peut permettre de compléter la solution **sprites** qui consiste à exploiter un seul fichier contenant plusieurs images de tailles différentes collées les unes aux autres et affichées en exploitant le **background-position** en css, exemple : [Fig. A]

Ce process ne pourrait-il pas s'appliquer a l'intégralité d'un site Internet et en optimiser le temps de chargement ?

Les nouveautés de **Windows Server 2016** pour les développeurs

• Patrice Lamarche
Développeur
Chausson Matériaux

Disponible depuis octobre 2016, trois ans après la mise à disposition de Windows Server 2012 R2, Microsoft nous propose une nouvelle version majeure de son système d'exploitation pour serveurs.

Une nouvelle version qui propose quelques nouveautés importantes pour les développeurs que nous allons parcourir ensemble.

Internet Information Services 10

Inclus dans Windows Server 2016 et disponible uniquement au sein de cette version de Windows, IIS 10 n'est pas une version majeure du serveur Web de Microsoft. IIS 10 ne propose donc pas d'importantes évolutions techniques ni une refonte d'architecture comme cela pouvait être le cas avec IIS 7. IIS 10 propose en effet une seule et unique nouveauté importante : le support d'http/2.

Le support d'http/2

Publié en Mai 2015, un petit peu moins de 20 ans après la publication d'http 1.1, cette nouvelle version d'http permet d'adapter le protocole aux usages du Web valables depuis déjà bien longtemps ; plusieurs nouveautés permettront à vos utilisateurs de bénéficier de meilleures performances.

La plus grosse nouveauté proposée par http/2 concerne la gestion des connexions TCP. Alors que la grande majorité des navigateurs limitaient le nombre de connexions simultanées à un même domaine à 6, http/2 propose une nouvelle gestion des connexions TCP. Ainsi au lieu de devoir patienter le temps que les connexions TCP initiées se terminent afin de pouvoir profiter d'une connexion dans ce pool de 6 connexions (ce qui peut être vraiment pénalisant sur les applications actuelles où il n'est pas rare d'avoir plusieurs dizaines de ressources http utilisées depuis une page), http/2 propose le multiplexage de connexion.

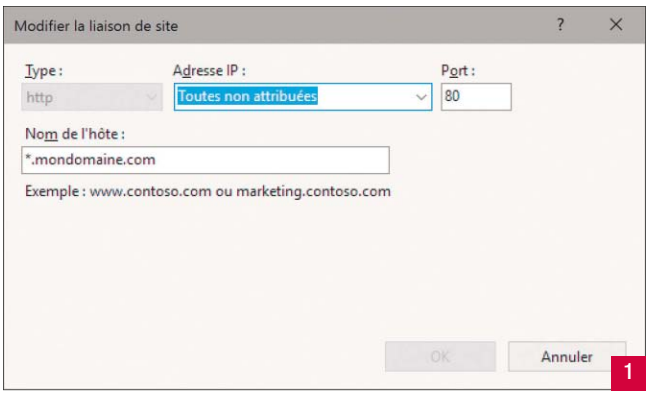
Une seule connexion TCP est ainsi établie, et le navigateur a à présent la possibilité de lancer un grande nombre de téléchargements de ressources (JS, CSS, etc.) en parallèle.

Cette simultanéité permet donc de profiter pleinement de la bande passante disponible lors de la consultation d'un site Web, de l'utilisation d'une application Web, sans être bloqué par cette limitation artificielle de 6 connexions simultanées.

Autre nouveauté, le support du push. Si vous souhaitez éviter d'attendre que le navigateur parse le contenu de votre page afin de démarrer le téléchargement des ressources nécessaires, vous avez la possibilité de pusher du contenu sans que le client ne le demande. Ainsi une fois le contenu de votre page parsée, le navigateur pourra puiser directement dans son cache, au lieu d'initier un nouveau téléchargement.

Cette possibilité est néanmoins à utiliser avec précaution afin d'éviter d'imposer le téléchargement d'un contenu qui, en réalité, n'est pas utilisée côté client.

Comme il vous appartient de décider des ressources que vous souhaitez mettre à disposition, il vous faut prêter attention à bien pusher les ressources pertinentes et utiles, afin que vos utilisateurs tirent bien un bénéfice de ce push, et ne soient pas au contraire pénalisés par cela.



1

Il est à noter qu'afin d'améliorer la sécurité de l'ensemble des applications Web, les navigateurs Web imposent l'utilisation de TLS afin de pouvoir bénéficier d'http/2. Si vous n'êtes pas encore passés sur HTTPS, vous devrez donc voir comment implémenter le chiffrement de connexions via TLS afin de pouvoir faire profiter vos utilisateurs de meilleures performances grâce au support d'http/2.

Le support des wildcards dans les bindings

Et enfin, dernière nouveauté mineure, mais qui peut s'avérer pratique, IIS 10 propose le support d'un binding générique pour vos sous-domaines. Ainsi, si vous souhaitez indiquer au serveur Web que les requêtes ciblant *.mondomaine.com doivent être redirigées vers tel site Web dans IIS, vous pouvez à présent le faire librement. [Fig.1]

Il sera donc à votre charge de gérer du côté de votre application ces sous-domaines afin de proposer un fonctionnel ou un contexte d'utilisation différent.

La gestion des conteneurs

Le support des conteneurs est très clairement la nouveauté la plus importante. Afin de ne pas perdre trop de parts de marché dans le secteur des serveurs et ne pas se laisser distancer technologiquement par les serveurs sous Linux, Microsoft a investi de manière importante afin d'être capable de proposer ce nouveau type de déploiement.

Pour réaliser efficacement cet investissement, l'éditeur s'est associé à Docker, afin de proposer gratuitement une implémentation native de celui-ci. Les équipes de Docker et de Microsoft se sont donc associées afin de développer en commun cette nouvelle fonctionnalité. Un partenariat win/win qui permet à Docker de continuer à s'imposer comme solution de conteneurisation leader presque incontournable, et à Microsoft de rattraper ce petit retard technologique sans investir dans le développement d'une technologie concurrente isolée.

Les conteneurs

Les services de conteneurisation sont souvent considérés comme des services de virtualisation de système d'exploitation. Ainsi contrairement aux machines virtuelles qui permettent de virtualiser des serveurs et qui virtualisent l'intégralité d'un serveur, couche hardware (CPU, RAM, disque durs, etc.) et système d'exploitation compris, les conteneurs permettent de déployer des applications au sein d'environnements d'exécution isolés, mais qui partagent la couche hardware de la machine hôte, et également le kernel du système d'exploitation de la machine hôte.

Il est ainsi possible de déployer un grand nombre de conteneurs au sein d'un serveur comparé à l'utilisation de machines virtuelles. Il est en effet possible de déployer plusieurs dizaines voire plus d'une centaine de conteneurs au sein d'une seule et même machine physique, alors que l'on atteint très facilement les limites hardware de la machine hôte ne serait-ce que d'un point de vue de la consommation de la RAM préemptée par chaque machine virtuelle.

En plus de cette densité plus importante qui permet d'utiliser plus efficacement des serveurs physiques, les conteneurs permettent également de bénéficier d'un environnement d'exécution isolé mais également indépendant des couches sous-jacentes.

Ainsi si vous souhaitez être capable de déployer votre application sur des serveurs on-premise ou sur le Cloud quel que soit le fournisseur de Cloud choisi (Microsoft Azure, Amazon, Google, OVH ou autre), les conteneurs peuvent représenter une solution pertinente qui vous simplifiera grandement la vie d'un point de vue réversabilité on-premise <-> Cloud ou encore réversibilité entre ces différents opérateurs Cloud.

Windows Serveur 2016 propose deux niveaux d'isolation d'exécution différents. Les conteneurs Windows sont l'équivalent des conteneurs que l'on peut retrouver sous Linux. Comme indiqué précédemment, le noyau du Windows de la machine hôte est partagé par l'ensemble des conteneurs en cours d'exécution.

Les conteneurs Hyper-V permettent quant à eux d'avoir un environnement encore plus isolé puisqu'avec ce niveau d'isolation, le kernel n'est plus partagé entre conteneurs, mais bien exécuté pour chaque conteneur. Comme leur nom l'indique, ces conteneurs nécessitent l'hyperviseur d'Hyper-V pour fonctionner ; une machine virtuelle légère spécialement conçue pour cette fonctionnalité est utilisée afin de permettre un démarrage de conteneur beaucoup plus rapidement qu'une machine virtuelle classique.

Les images

Les conteneurs sont basés sur des images qui contiennent tout le nécessaire pour exécuter nos applications. En premier lieu, les images contiennent une image du système d'exploitation utilisé. Il peut paraître surprenant d'encapsuler une image de système d'exploitation alors que le kernel de celui-ci est partagé entre la machine hôte et l'ensemble des conteneurs en cours d'exécution. Cette incorporation permet de s'assurer de bénéficier d'un environnement strictement identique quelle que soit votre machine hôte. Ainsi les drivers spécifiques à votre serveur ou les modifications qui peuvent être apportées par les constructeurs sont ignorées et ne sont pas chargés par les conteneurs.

Microsoft propose deux images de base pour Windows. Une contenant Windows Server Core 2016, et un autre contenant une nouvelle édition de Windows : Windows Nano Server.

Cette nouvelle édition ultra-light de Windows est 10 fois plus légère que Windows Server Core. On passe ainsi d'une image de 300 Mo a contrario d'une image de près de 4 Go pour une image Windows Server Core. Une édition donc idéale pour créer des conteneurs car beaucoup plus légère en termes de consommation mémoire, mais également plus économe en temps de démarrage.

Cette édition étant beaucoup plus légère, il n'est pas possible de faire fonctionner des applications basées sur le framework .net. Dans le monde .net, seules les applications .net Core sont supportées.

En plus du système d'exploitation, une image contient le framework dont votre application dépend, et bien évidemment votre application ainsi que ses autres dépendances applicatives.

Microsoft propose plusieurs images afin de simplifier la création d'images pour déployer des applications .net. Il n'est donc pas utile de partir d'une image de base pour recréer une image « from scratch » ; vous pouvez tout à fait partir d'une image prête à l'emploi. [Fig.2]

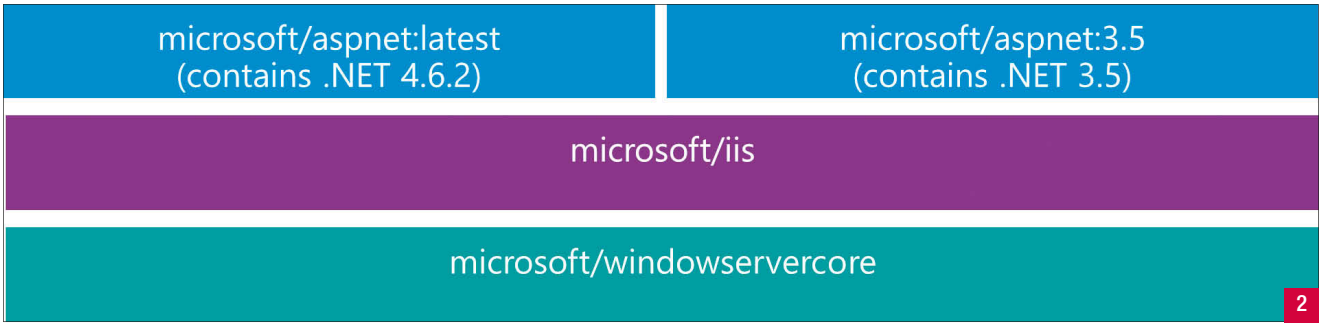
En pratique

Une image se construit à partir d'un fichier dockerfile. Ce fichier permet d'indiquer différentes commandes qui permettent de construire l'intégralité de votre environnement d'exécution. De la définition de votre image de base, à l'exécution de toutes les commandes nécessaires au déploiement de votre application.

Ainsi pour une application ASP.net Webforms ou MVC, vous pouvez créer un fichier dockerfile en suivant cet exemple :

```
FROM microsoft/aspnet
RUN mkdir C:\Demo
# configure le site dans IIS
RUN powershell -NoProfile -Command \
Import-module IISAdministration; \
New-IISSite -Name « ASPNET » -PhysicalPath C:\Demo -BindingInformation « *:8000: »
# Indique au conteneur d'écouter sur le port 8000
EXPOSE 8000
# Copie les fichiers du dossier courant dans le dossier demo
ADD . /demo
```

Il est important de comprendre que chaque commande spécifiée dans



un fichier docker file crée une couche. Les images Docker sont en effet basées sur un principe de couche très intéressant lors de la gestion de vos images. Ainsi lors de la récupération d'une image, seules les couches non disponibles sur votre machine hôte seront téléchargées. Vous ne téléchargerez donc pas plusieurs Go pour chaque récupération d'image basée sur Windows Server Core.
Afin de créer l'image, il est nécessaire de lancer une invite de commande (cmd ou powershell) et de lancer la commande suivante :

```
docker build –t demo .
```

Et enfin pour lancer un conteneur basé sur cette image :

```
docker run –name demo –d –p 8000:8000 demo
```

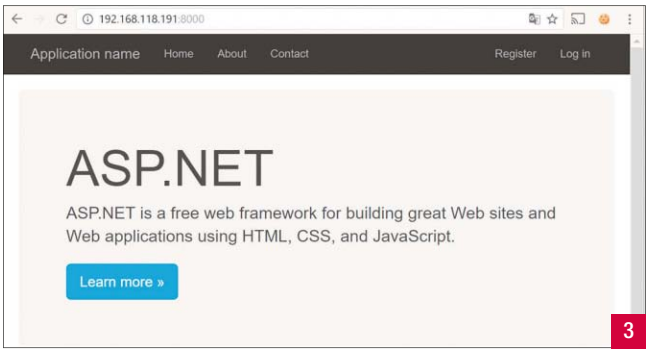
Cette commande permet d'exécuter un conteneur en lui donnant un nom, et en spécifiant le mapping de port entre la machine hôte et le conteneur.
A noter que Windows ne permet pas d'accéder à des applications conteneurisées en utilisant l'adresse de loopback localhost. Il vous faut donc contourner cette limitation en récupérant l'adresse ip du conteneur pour accéder à votre application depuis votre machine hôte. Vous pouvez récupérer celle-ci en exécutant la commande suivante :

```
docker inspect –format= »{{.NetworkSettings.Networks.nat.IPAddress}} » demo
```

Vous pourrez ainsi accéder à votre application depuis votre navigateur en spécifiant l'adresse ip trouvée et le port spécifié lors du démarrage du conteneur. [Fig.3]

Développer avec Docker et Visual Studio

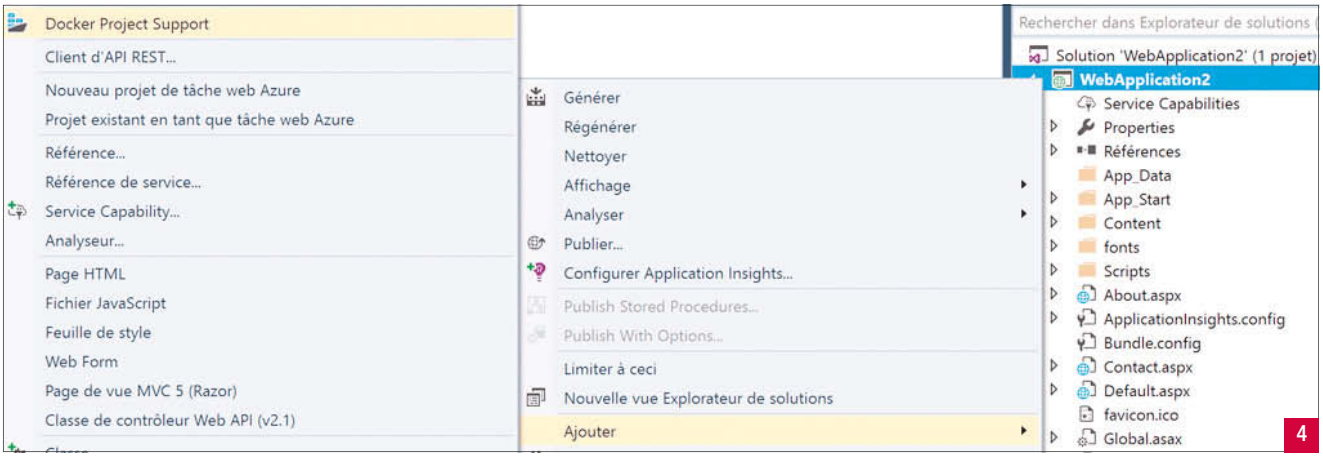
Microsoft propose des extensions pour Visual Studio qui permettent de simplifier la création du fichier dockerfile qui lui-même permet de définir le contenu d'une image (et qui permet de déployer et débuguer des applications au sein de conteneurs).
Bien que les Visual Studio Tools for Docker soient disponibles pour Visual Studio 2015, il est préférable d'utiliser Visual Studio 2017 pour profiter d'une intégration bien plus complète. Applications ASP.net Webforms ou MVC basées sur le framework .net sont ainsi supportées en plus du développement d'applications ASP.net Core basées sur le framework .net Core.



Une fois ces extensions installées, vous aurez donc la possibilité de bénéficier de la simplicité appréciée de Visual Studio : un simple F5 et vous voici en train de déboguer votre application déployée au sein d'un conteneur Docker ! [Fig.4]

Une sortie tardive

Le géant de Redmond a donc mis 3 ans avant de proposer une nouvelle version de Windows Server. Un délai important, probablement même trop important, dans une industrie informatique prônant l'agilité et les raccourcissements des rythmes de livraison.
Il n'est pas question de contester la charge de travail nécessaire pour créer une nouvelle édition Windows Server telle que Windows Nano Server. Refactorer des millions de ligne de code afin de ré-architecturer le système d'exploitation et briser un grand nombre de dépendances représente une tâche dantesque souvent sous-estimée et mal appréciée.
La modification de Windows afin d'ajouter de nouvelles primitives permettant de proposer une intégration de Docker au sein de Windows est également un investissement important, et un développement réalisé en étroite collaboration avec les équipes de ce nouveau partenaires.
Il est surprenant de constater que le géant de Redmond n'ait pas souhaité décorréler le cycle de vie de son serveur Web Internet Information Services de celui de Windows.
Conséquence lourde, les développeurs Web utilisant des technologies telles qu'ASP.net Webforms ou ASP.net MVC ont dû patienter 1 an de plus que les développeurs utilisant Apache afin d'être capables d'exploiter le nouveau standard http/2.
Il va donc être intéressant de voir si Microsoft va modifier sa stratégie de livraison suite à la mise à disposition de cette nouvelle édition.



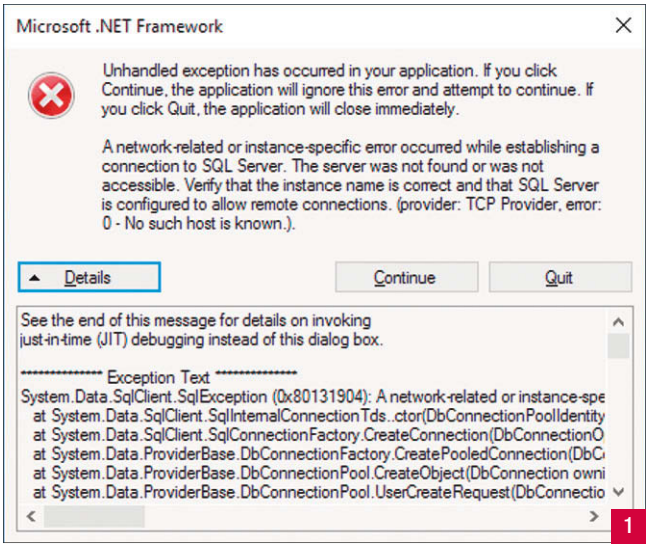
Le debugging sous Windows

Christophe PICHAUD
.NET Rangers by Sogeti
Consultant sur les technologies Microsoft
christophepichaud@hotmail.com
www.windowscpp.net

Depuis que les développeurs font du logiciel, il est une pratique qui n'a cessé d'être le meilleur ami ou ennemi du développeur : le debugging. On parle du terme en anglais plutôt que du terme « mise au point ». Le debugging est un art comme une enquête policière. En effet, on cherche des pistes, on a des indices et des suspicions… On s'en relève la nuit ou on en rêve… Et quand on trouve le ou les bugs, on reçoit un flux émotionnel de soulagement. Bref, voilà la trame. Nous allons évoquer plusieurs scénarios sur du code .NET avec un outil du monde natif (C/C++) : le debugger système WinDBG.

En .NET Framework

L'avantage du développement en .NET Framework c'est que l'exception que nous cherchons a de bonnes chances d'être captée par le runtime, et une stack trace nous est proposée en appuyant sur le bouton détail. Dans mon cas, il s'agit d'une chaîne de connexion qui pointe sur un serveur que je n'ai pas… Bref vous allez me dire « c'est super simple le debugging .NET ». Oui vu comme ça pour une application GUI… [Fig.1].



1

Par contre, cela peut vite devenir invivable si la boîte s'affiche toutes les 10 secondes car cela veut dire que le mécanisme de gestion des exceptions n'a pas été mis en place dans l'application.

La méthode à Papa

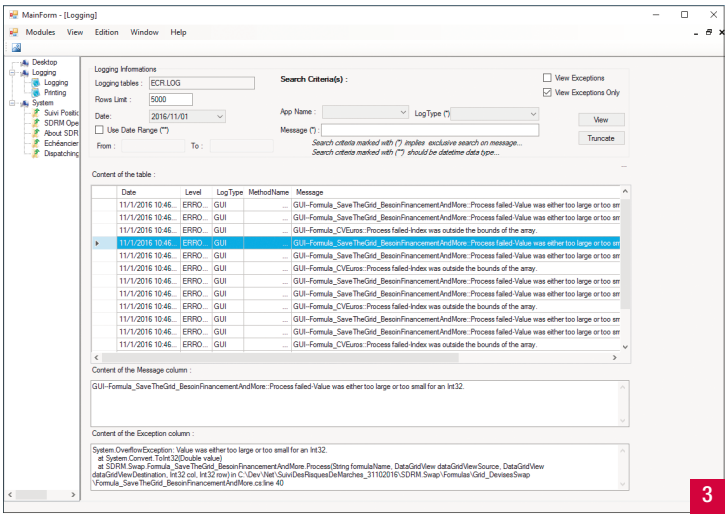
C'est une méthode que j'utilise depuis 20 ans et qui marche à tous les coups. Par contre c'est peut-être un bad-pattern ou un real-life pattern : faut voir…[Fig.2]. Le principe est le suivant : chaque méthode business possède au minimum un handler d'exception globale et au minimum un log est produit dans un fichier. Je suis un fan de log4net, donc dans mon cas, le log va dans la console de debug, dans des fichiers tournants (RollingFileAppender) et en base de données. Vous allez me dire, en base ? Quel est l'intérêt ? J'ai un outil de debugging qui me permet de faire des recherches dans les traces effectuées en bases, et c'est bien pratique quand des dizaines de routines business tournent en même temps. [Fig.3]. Cet outillage et cette manière de gérer les traces n'est pas valable que pour les applications graphiques. C'est aussi valable pour les services Windows ou le développement serveur.

A retenir

Il vaut mieux mettre des traces pendant le développement plutôt que de chercher aléatoirement un bout de code qui soit susceptible de planter. L'avantage du logger log4net c'est qu'il est possible de le mettre hors de service sans avoir à recompiler son code. Un simple switch dans le fichier de configuration permet d'ajuster le niveau de traces.

```
class Formula_StopLoss_Gains : IFormula
{
    62 references
    public void Process(string formulaName,
                        DataGridView dataGridViewSource,
                        DataGridView dataGridViewDestination,
                        int col, int row)
    {
        try
        {
            if (col == Convert.ToInt32(ColEnum.Devise_1)
                || col == Convert.ToInt32(ColEnum.Devise_2)
                || col == Convert.ToInt32(ColEnum.TakeProfit)
                || col == Convert.ToInt32(ColEnum.MontantDevise1EnKDevise)
                || col == Convert.ToInt32(ColEnum.CoursPosition))
            {
                //
                // Some Work ...
                //
            }
        }
        catch (Exception ex)
        {
            Logger.Error("Formula_StopLoss_Gains::Process failed", ex);
        }
    }
}
```

2



3

Comment faire sa routine de log ?

Voici un concentré de la DLL qui supporte log4net. Comme vous allez le voir c'est très simple à utiliser :

```
using log4net;
using log4net.Appender;
using log4net.Config;
using log4net.Core;
using log4net.Layout;
using log4net.Repository;
using log4net.Repository.Hierarchy;

namespace MyLibrary
{
  public class Log
  {
    private static readonly object LockingObject = new object();
    private static Log logInstance;
    private static ILog internalLog;

    private Log()
    {
    }

    public static Log Instance
    {
      get
      {
        if (logInstance == null)
        {
          lock (LockingObject)
          {
            if (logInstance == null)
            {
              logInstance = new Log();
            }
          }
        }

        return logInstance;
      }
    }

    internal string LogPath { get; private set; }
    private string EnvironmentPath { get; set; }

    public bool InitializeUserLog(string environmentPath, string fileName)
    {
      try
      {
        this.EnvironmentPath = environmentPath;
        this.LogPath = this.EnvironmentPath;
        if (Directory.Exists(this.LogPath) == false)
        {
          Directory.CreateDirectory(this.LogPath);
        }

        return this.InitializeAppenders(fileName);
      }
      catch (Exception ex)
      {
        System.Diagnostics.Trace.WriteLine("Unhandled exception " + ex.Message + "\n" + ex.StackTrace);
        return false;
      }
    }

    public void Debug(string msg)
    {
      if (internalLog.IsDebugEnabled)
      {
        this.Logger(Level.Debug, msg);
      }
    }

    public void Error(string msg, Exception ex = null)
    {
      if (internalLog.IsErrorEnabled)
      {
        this.Logger(Level.Error, msg, ex);
      }
    }

    public void Info(string msg)
    {
      if (internalLog.IsInfoEnabled)
      {
        this.Logger(Level.Info, msg);
      }
    }

    public void Warning(string msg)
    {
      if (internalLog.IsWarnEnabled)
      {
        this.Logger(Level.Warn, msg);
      }
    }

    private bool InitializeAppenders(string fileName)
    {
      // Create the RollingFileAppender
      RollingFileAppender rla = new RollingFileAppender();
      rla.Name = "MyRollingFileAppender";
      rla.File = string.Format("{0}\\{1}_{2}.log", this.LogPath, fileName, Environment.UserName);
      rla.DatePattern = "yyyyMMdd";
      rla.MaximumFileSize = "1MB";
      rla.MaxSizeRollBackups = 10;
      rla.Threshold = Level.Debug;
      PatternLayout layout = new PatternLayout();
      layout.ConversionPattern = "[%date] - %-5level - %message%newline";
      layout.ActivateOptions();
      rla.Layout = layout;
      rla.ActivateOptions();
```

```
        // Create the ConsoleAppender
        ConsoleAppender ca = new ConsoleAppender();
        ca.Name = "MyConsoleAppender";
        ca.Threshold = Level.Debug;
        ca.Layout = new SimpleLayout();
        ca.ActivateOptions();

        BasicConfigurator.Configure(new IAppender[] { rla, ca });

        // Set the Root logger
        Hierarchy hierarchy = (Hierarchy)LogManager.GetRepository();
        Logger logger = hierarchy.Root;
        logger.Level = Level.Debug;

        // Set internal logger
        internalLog = LogManager.GetLogger(typeof(Log));

        return internalLog != null;
    }

    private void Logger(
        Level level,
        string message,
        Exception exception = null)
    {
        LoggingEvent loggingEvent = new LoggingEvent(typeof(Log), internalLog.Logger.
Repository, internalLog.Logger.Name, level, message, exception);
        internalLog.Logger.Log(loggingEvent);
    }
  }
}
```

Avec ce bout de code qui contient une classe de Log, vous pouvez l'utiliser partout !

```
public bool InitSession()
{
  try
  {
    Log.Instance.Info("InitSession...");

    //
    // Do some work ...
    //

    return true;
  }
  catch (Exception ex)
  {
    Log.Instance.Error("InitSession failed", ex);
    return false;
  }
}
```

Remarquez comment est construite cette routine. Elle renvoie un booléen. On a une information de log à l'entrée de la fonction, et une information s'il y a eu une exception. J'aime bien cette construction. Le debugging sera très facile car on a préparé le terrain… Et puis avouons-le, cela ne coûte rien mais pourtant, personne ne le fait… C'est dommage. Au passage, ce qui est fait ici en C# peut l'être aussi en C++. La librairie log4cpp s'utilise et se configure (presque) comme log4net. L'avantage du code .NET c'est que l'on a une stack trace directement exploitable alors qu'en natif il faut provoquer soi-même la construction de la stack trace via StackWalk, une API Win32. La classe C++ qui encapsule cette possibilité native est disponible dans (accrochez-vous) le Microsoft System Journal de 1997 par Matt Pietrek et sa classe MSJExceptionHandler. Ok, on laisse les cadavres à la cave…

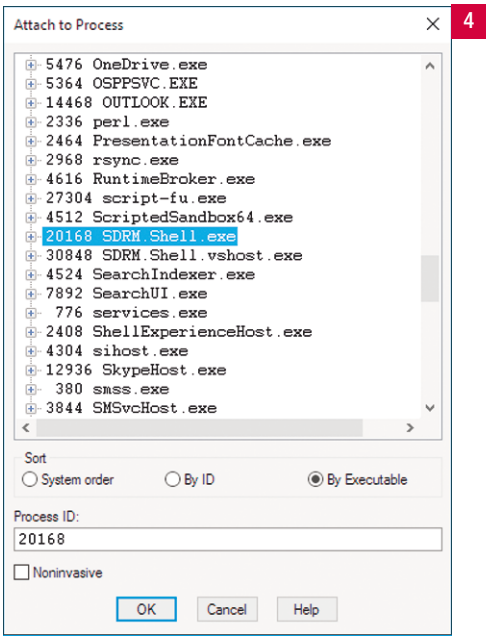
Débugger sans les sources

Quoi ? Vous allez me dire qu'est-ce que c'est que ce délire ? Debugger sans les sources ? Eh oui mon petit bonhomme… Juste avec les fichiers PDB. Vous vous souvenez que dans votre répertoire bin, il y a vos exe & dll, mais aussi des fichiers PDB. Ces fichiers sont les informations symboliques de debugging. Imaginons le scénario suivant : le serveur est mis en défaut de temps en temps et les équipes de dev n'arrivent pas à reproduire le problème. Je sens que là je vais en intéresser plus d'un ! Dans ce cas, il y a souvent un grand moment de solitude car c'est le moment où souvent, les managers rentrent dans la boucle, les emails partent, la situation se tend, et l'ambiance en prend un coup. Ce n'est pas le moment de sortir un épisode des lapins crétins ou bien une blague du commitstrip... Dans cette situation, pas la peine de chercher sur stackoverflow ; c'est à votre système d'exploitation de faire le taf. Oui votre OS Windows. On va lui demander de catcher tout seul, comme un grand, l'exception que l'on cherche, et qui n'arrive que de temps en temps… Votre ami se nomme WinDBG et c'est le debugger Windows officiel. WinDBG est distribué dans le Windows SDK ou le Windows DDK. Oui je sais, depuis que Visual Studio prend tout en charge, vous n'installez plus le SDK ou le DDK : c'est triste. Bref, téléchargez le Windows SDK et installez l'outil WinDBG.

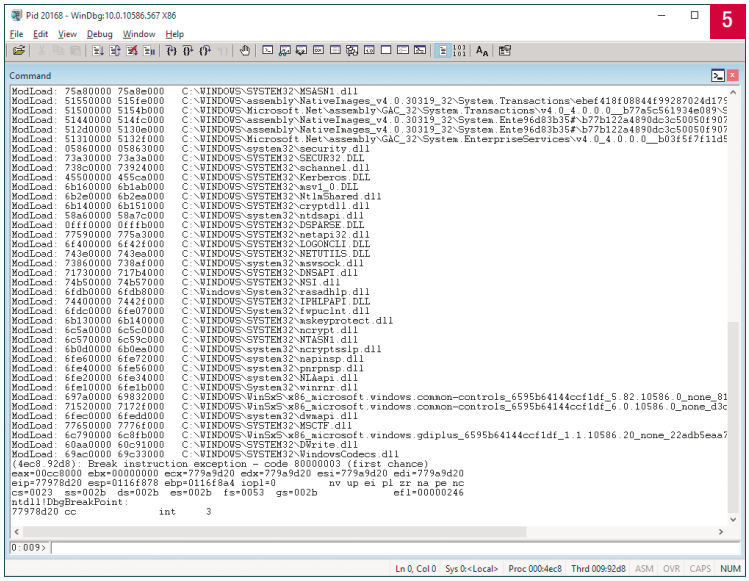
WinDBG : le debugger système

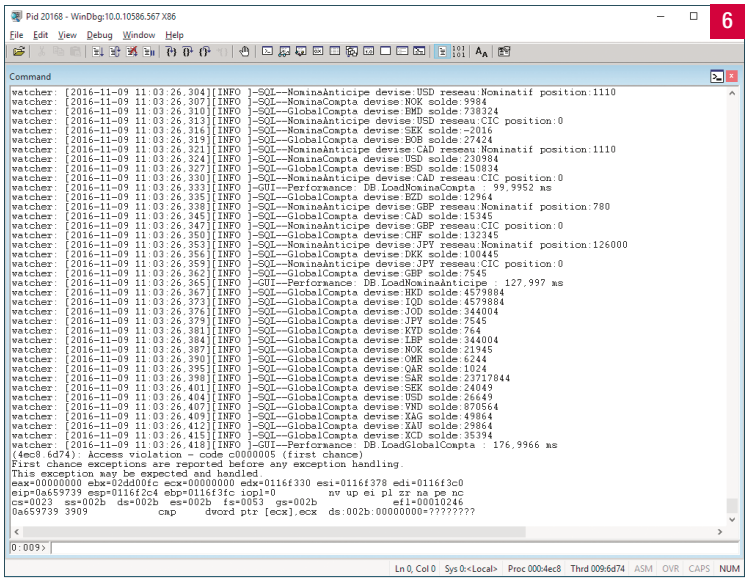
L'avantage de WinDBG c'est qu'il sait tirer parti de suite des fichiers PDB. Vous le lancez et vous demandez à vous « attacher » à un processus. Choisissez le processus par nom ou par ID et voici ce que WinDbg affiche. [Fig.4]. Appuyez sur OK et le WinDBG va provoquer l'arrêt en breakpoint de l'application attachée. [Fig.5].

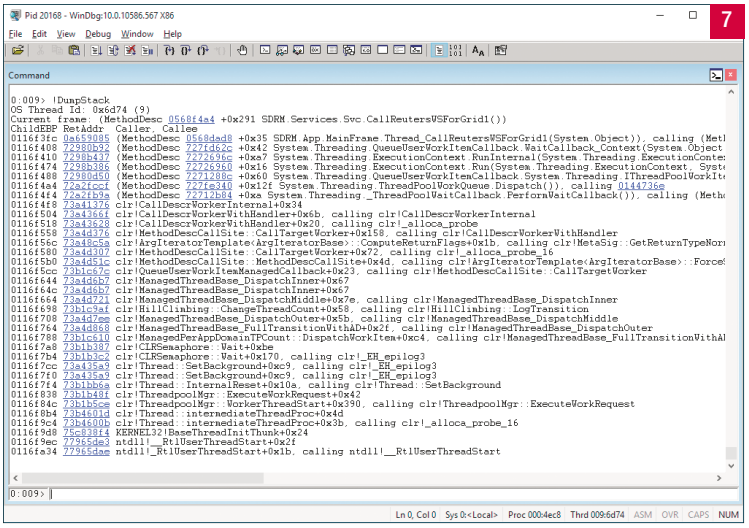
Il faut maintenant relancer l'application, car elle est à l'arrêt. Notre stratégie est simple : on laisse tourner l'application jusqu'au moment où l'on

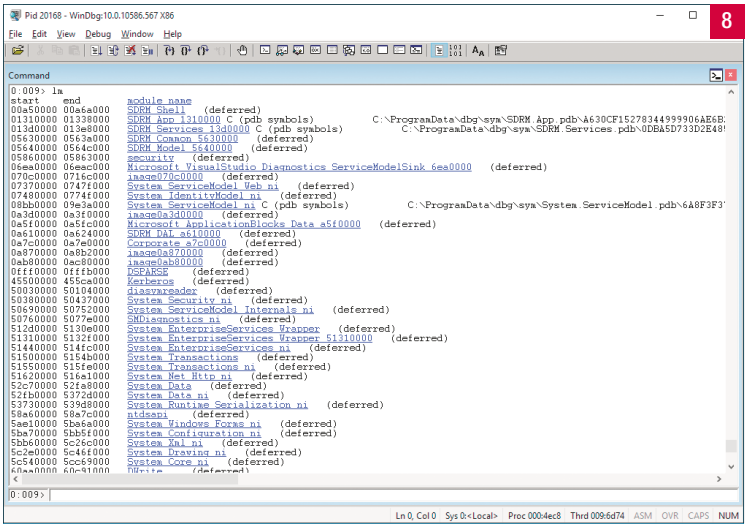


4









rencontre une exception. Lancez la commande « **g** » pour go. Après plusieurs actions faites dans mon application SDRM, WinDBG stoppe l'application car un plantage (pas une exception) est survenu. Voici l'aperçu de WinDBG : [Fig.6]. Vous remarquerez que mon logger log4net crash les logs dans le debugger. C'est très important de le faire avant « les plantages »… Cela réclame un effort qui permet d'être dans une zone de confort ensuite pour le debugging avancé.

A ce moment-là, il faut charger « Son of Strike » alias SOS.DLL pour obtenir des informations sur le code managé car l'application est faite en C# / .NET. Lancez la commande suivante :

```
.load C:\Windows\Microsoft.NET\Framework\v4.0.30319\sos.dll
```

Tapez aussi **!help** et vous aurez les commandes disponibles.

Maintenant que WinDBG a arrêté le processus suite à une exception non gérée, il faut obtenir une stack des appels réalisés pour savoir d'où vient le problème. Lancez la commande suivante :

```
!DumpStack [Fig.7]
```

La callstack présente aussi bien le code du CLR (Common Language Runtime) que le code de mon application SDRM. Le debugger est très clair : l'exception est dans la méthode *SDRM.Services.SVc.CallReutersWSForGrid10*. L'avantage de WinDBG est qu'il est efficace : dès qu'un plantage de type « Access Violation » survient, il met le processus en pause. Nous pouvons le relancer avec la commande « g » mais il s'arrêtera de nouveau si le plantage survient de nouveau. Vous remarquerez que le code de SDRM est avec les noms de méthodes. Vous pouvez vérifier que les symboles de debug sont chargés en exécutant le command « **lm** ». [Fig.8]

CONCLUSION

Pour debugger sur le serveur ou sur un poste de production, il est facile de mettre des atouts de son côté pendant le développement:

- Utiliser le namespace System.Diagnostics et les méthodes Debug() ;
- Utiliser un logger intelligent comme log4net et activez plusieurs options (console, mémoire, fichiers, db) ;
- Gérer des exceptions dans les méthodes business ;
- Installez WinDBG sur le serveur ;
- Installez ILSpy car parfois on ne se rappelle pas de tout ce que fait le code. De plus, ça peut donner des orientations sur la stratégie de debugging.

L'utilisation de WinDBG avec sos.dll est très bien documentée sur le site MSDN. Si vous êtes curieux, vous pouvez refaire toutes les opérations que fait Visual Studio en mode debug depuis WinDBG en tapant des commandes à la main. •

Développer avec des composants : l'exemple de Kendo UI

Carole CHEVALIER
Ingénieur Concepteur .NET chez **SOAT**

Les librairies de composants d'interface Web se présentent comme des boîtes à outils dans lesquelles on peut piocher des widgets permettant de simplifier le code des pages Web, voire de designer des pages avec très peu de lignes de code. Utiliser des composants pour développer ses applications Web ou mobiles est tentant par la facilité de prise en main et l'économie en temps de développement avancés par les éditeurs sur leurs sites Internet.

Plusieurs frameworks sont disponibles sur le marché avec différentes licences : gratuites, payantes ou semi-payantes. Certains sont basés sur JQuery, ce qui permet de les utiliser sur tous les projets Web et mobile, sans problème de compatibilité avec le fameux framework Javascript. On pense immédiatement à JQuery UI(1), mais nous verrons en particulier Kendo UI(2) de l'éditeur Telerik, et son wrapper pour ASP.NET MVC. Ignite UI(3), de l'éditeur Infragistics, propose le même genre de fonctionnalités que Kendo UI / ASP.NET MVC. Nous ferons dans cet article un petit tour d'horizon des fonctionnalités proposées par les éditeurs et de quelques avantages et lacunes de ces librairies.

Kendo UI/MVC

Kendo UI est un framework HTML5 et Javascript basé sur JQuery qui met à disposition de nombreux widgets ou des solutions pour développer des applications Web modernes. Deux versions du framework sont disponibles :

- **Kendo UI Core** est une version libre et open source distribuée sous licence Apache version 2.0., contenant 24 widgets open source, et 12 supplémentaires qui sont destinés à la programmation mobile.
- **Kendo UI Professionnel** est une version complète du framework et sous licence commerciale, proposant 18 widgets supplémentaires et un support professionnel.

Il existe des wrappers pour ASP.NET MVC mais aussi pour JSP, PHP, ainsi que pour le développement natif mobile ou Xamarin. Les exemples de cet article utiliseront des wrappers pour ASP.NET MVC, qui sont sous la forme de HTMLHelpers et permettent de gérer plus facilement les fonctionnalités serveur des widgets.

La Grid Kendo

Un exemple de composant parmi les plus populaires est la Grid : une grille de données ultra souple que l'on peut configurer pour afficher telle ou telle propriété d'une liste d'objets et effectuer entre autres des tris, filtres, groupements poussés de données.

Plusieurs modes de bindings sont disponibles pour la Grid Kendo. Le **binding côté serveur** est celui défini par défaut, il effectue des requêtes GET et POST lors d'actions sur la grille comme l'édition, la suppression, ou des actions de tri ou de filtrage, par exemple.

```
@(Html.Kendo().Grid((IEnumerable<Programmez.Example.Models.MyViewModel>)
ViewData["products"])
        .Name("Grid")
        .Columns(columns =>
        {
            columns.Bound(p => p.Id);
            columns.Bound(p => p.Name);
        })
)
```

Une autre syntaxe est possible avec la méthode BindTo :

```
@(Html.Kendo().Grid<Programmez.Example.Models.MyViewModel>()
      .Name("Grid")
      .BindTo((IEnumerable<Programmez.Example.Models.MyViewModel>)ViewBag.Products)
      .Columns(columns =>
      {
          columns.Bound(p => p.Id);
          columns.Bound(p => p.Name);
      })
)
```

Avec le **binding Ajax**, lesdites requêtes seront faites de manière asynchrone et la page ne sera pas rechargée intégralement. La grille recevra un objet JSON lui permettant de se mettre à jour, et toutes les actions sur la grille se font côté client. Il est possible de faire un gros travail de customisation de la grille en s'abonnant aux événements Javascript de ces actions et en utilisant des templates Razor pour afficher les contenus.

```
@(Html.Kendo().Grid<Programmez.Example.Models.MyViewModel>()
        .Name("Grid")
        .DataSource(dataSource => dataSource
            .Ajax()
            .Read(read => read.Action("Get", "Grid"))
        )
        .Columns(columns =>
        {
            columns.Bound(product => product.Id);
            columns.Bound(product => product.Name);
        })

)
```

Ces deux modes peuvent être customisés pour que la grille effectue des tâches personnalisées de tri ou de filtrage sur la grille en activant l'option EnableCustomBinding.

(1) https://jqueryui.com/
(2) http://www.telerik.com/kendo-ui
(3) http://www.igniteui.com/

Des fonctionnalités avancées

La Grid permet d'implémenter très facilement du tri, des filtres, et des groupements sur les colonnes, ainsi qu'une pagination très souple. Il suffit d'activer ces options et de les configurer en utilisant les méthodes Filterable, Pageable et Sortable sur la grille. Les tris et filtres peuvent être également activés colonne par colonne. L'édition peut se faire en mode Inline (ligne par ligne), Batch (édition simultanée de toute les lignes à l'écran) ou Popup (petit formulaire qui s'affiche dans une fenêtre interne). Un paramètre de la méthode Editable permet de passer de l'un des modes à un autre.

```
@(Html.Kendo().Grid<Programmez.Example.Models.MyViewModel>()
    .Name("Grid")
    .Columns(columns =>
    {
        columns.Bound(p => p.Id).Width(150);
        columns.Bound(p => p.Name);
        columns.Command(command => { command.Edit(); command.Destroy();
}).Width(250);
    })
    .ToolBar(toolbar => toolbar.Create())
    .Editable(editable => editable.Mode(GridEditMode.InLine))
    .DataSource(dataSource => dataSource
        .Ajax()
        // définition de la clé
        .Model(model => model.Id(p => p.Id))
        // définition des actions mvc pour chacune des actions sur la grille
        .Create(update => update.Action("Create", "Grid"))
        .Read(read => read.Action("Read", "Grid"))
        .Update(update => update.Action("Update", "Grid"))
        .Destroy(update => update.Action("Destroy", "Grid"))
    )
)
```

Il est aussi possible de hiérarchiser plusieurs grilles entre elles, grouper les données par colonnes, exécuter des actions lors de la sélection d'une cellule ou d'une ligne, agréger les données des lignes ou des colonnes, exporter sous forme de fichier Excel et PDF, bref, toutes les fonctionnalités qu'on peut attendre d'une grille riche.

Documentation et rapidité de mise en place

Utiliser des composants pour développer son application octroie un **gain de temps** non négligeable en termes de mise en place d'un projet et de développement. Les librairies de composants sont très largement documentées sur le site Web des éditeurs : **description de l'API**, **tutoriaux** et nombreux **exemples d'implémentation** faciles d'accès ainsi que leurs variantes possibles permettent de trouver facilement quelle fonctionnalité de quel composant sera utile à l'application. Des **exemples complets** d'applications utilisant d'autres frameworks comme Bootstrap(4) ou AngularJS(5) servent d'appui dans les débuts, et des environnements de test tels que le Page Designer d'Infragistics ou le Dojo de Telerik avec des tutoriaux intégrés permettent de se former rapidement au fonctionnement des modules.

Un framework est **rapidement pris en main** et l'application se monte à toute vitesse avec peu de lignes de code.

(4) http://getbootstrap.com/
(5) https://angularjs.org/
(6) http://jqueryui.com/themeroller/
http://demos.telerik.com/kendo-ui/themebuilder/

De larges possibilités avec un seul système

L'un des avantages des bibliothèques de composants comme Kendo est qu'elles proposent un **choix conséquent de widgets et de solutions**, ce qui évite de multiplier les librairies sur une même application et les problèmes de **compatibilité**, que ce soit de code (compatibilité des scripts Javascript) ou de thème visuel, qui apparaissent parfois quand on en cumule plusieurs. Les **thèmes variés** de ces librairies s'intègrent facilement à une charte graphique existante : Kendo UI en propose une quinzaine. Ignite UI est moins généreux mais propose des thèmes s'intégrant spécifiquement dans les interfaces iOS, Metro, ou Bootstrap. La palme revient à JQuery UI qui propose 24 thèmes très variés. Il est même possible de customiser les thèmes existants grâce à une interface Web sur le site des éditeurs(6). Les composants disponibles couvrent une très large part des interfaces visuelles nécessaires dans une application : gestion et affichage de **données** (grille, tableur, liste arborescente…), éléments de **formulaire** (combobox, éditeurs WYSIWYG, upload asynchrone de fichiers, validation de données…), **navigation** (menus, barres d'outils, boutons.), **graphiques** (en colonnes, en camembert, radar, codes barre et QRCode, cartes, …) et diverses **interfaces responsives** (fenêtres internes, tooltip, gestion des notifications…). De nombreux événements Javascript sont exploitables et permettent d'augmenter **l'interactivité** des composants : au passage de la souris ou au clic sur une série d'un graphe, au chargement des données d'une Grid ou d'une DropDownList, au clic sur un bouton de la barre d'outil d'une fenêtre interne, etc.

Un gros avantage des librairies Javascript de manière générale est la **compatibilité avec une multitude de navigateurs**. Les librairies de composants basés sur JQuery ne font pas exception. Certaines fonctionnalités très spécifiques des graphes ou certaines animations nécessitent une version d'Internet Explorer supérieure à 9 voire 10, mais dans l'ensemble, les widgets sont très bien supportés par les principaux navigateurs du marché : Internet Explorer, Edge, Firefox, Chrome, Opera, Safari, ...

Des widgets performants

Que les bibliothèques soient composées d'un nombre conséquent de modules n'est cependant pas synonyme de lenteurs. Certains composants, notamment ceux de visualisation de données, sont susceptibles d'être gourmands en temps de chargement, puisqu'ils sont destinés à être alimentés de plusieurs centaines, milliers, voire dizaines de milliers de lignes de données. Des **efforts conséquents** ont été consentis par les éditeurs pour optimiser au maximum le chargement et le traitement de ces composants. La **virtualisation** des données, en définissant à l'avance combien de lignes (et/ou de colonnes, chez Ignite UI) à afficher à l'écran, permet de réutiliser les noeuds du DOM existants du composant pour montrer plus de données par action de scroll. C'est une sorte de **pagination virtuelle** qui n'affiche qu'une partie des lignes à afficher, en laissant une expérience fluide à l'utilisateur.

Les données peuvent être au choix complètement chargées par le client, ou bien la pagination peut être gérée par le serveur avec Kendo UI en utilisant les options serverPaging et serverSorting de la source de données.

Le wrapper ASP.NET MVC de Kendo permet d'activer la virtualisation très simplement en utilisant la méthode Scrollable.

```
@(Html.Kendo().Grid<Programmez.Example.Models.MyViewModel>()
    .Name("Grid")
    .Columns(columns =>
    {
        columns.Bound(o => o.Id).Width(50);
```

```
        columns.Bound(o => o.Name).Width(130);
    })
    .Sortable()
    .Scrollable(scrollable => scrollable.Virtual(true))
    .HtmlAttributes(new { style = "height: 540px;" })
    .DataSource(dataSource => dataSource
        .Ajax()
        .PageSize(100) // nombre de lignes affichées à l'écran
        .Read(read => read.Action("Get", "Grid"))
    )
)
```

La plupart des applications front-end nécessitent d'implémenter des templates performants. Kendo UI intègre un système de **micro-templating** Javascript à l'exécution particulièrement rapide, et annoncé 60 fois plus performant que celui de JQuery Templates(7).
La syntaxe inline est particulièrement simple :

```
var t = kendo.template("<div><#= name #></div>");
```

Une autre syntaxe est prévue pour interpréter du HTML :

```
var t = kendo.template("<div>${ name }</div>");
```

Les templates et expressions réutilisables peuvent être définis de la manière suivante :

```
<script id="myTemplate" type="text/x-kendo-template">
    <ul>
        <# for (var i=0; i < data.length; i++) { #>
            <li><#= data[i] #></li>
        <# } #>
    </ul>
</script>
```

Toutefois, de manière générale, les librairies ne sont pas fermées à l'utilisation d'autres moteurs de templates si c'est nécessaire dans l'application. Les librairies de templating Javascript externes, comme Mustache(8) ou Handlebars(9) par exemple, bénéficient d'une bonne intégration avec Kendo. Ignite UI permet de créer des templates conditionnels avec TypeScript(10) et donne des solutions pour créer des bridges entre ces librairies et son propre moteur. La librairie Kendo propose avec **DataViz** tout un choix de composants qui permettent de visualiser des données sous forme de graphes, jauges ou encore codes barres et QRCodes, en utilisant des technologies modernes comme SVG(11 et Canvas(12) pour leur génération. La compatibilité avec d'anciens navigateurs est gérée en basculant sur du VML(13) si nécessaire. Des animations et des fonctions d'agrégations, telles que des moyennes, permettent l'affichage de données complexes sur des graphes élégants et performants.

Souplesse de chargement

Les librairies de composants sont donc performantes malgré la variété de composants proposée, mais il n'est pas nécessaire d'importer tout un framework pour n'en utiliser qu'une partie : il est souvent possible de **composer** ses packages au cas par cas avec seulement les plugins qui nous sont utiles, soit en chargeant les modules manuellement, soit en utilisant un outil qui sélectionne les scripts à charger dans la page où ils sont requis. Ignite UI met à disposition un **loader**(14) qui permet de charger à la demande les packages qui sont utiles à la page et au moment où ils sont nécessaires. Les scripts de Kendo UI se présentent quant à eux comme des **modules AMD**(15). Il est donc possible de choisir d'utiliser selon ses préférences un loader Javascript compatible, comme RequireJS(16) ou SystemJS(17), ce qui permettra de ne charger que les modules nécessaires plutôt que tout le fichier kendo.all.min.js et ainsi fluidifier le chargement de la page. Les bons outils pour le bon usage. De manière générale, les composants Web des éditeurs sont parfaits dans les **cas précis** pour lesquels ils ont été conçus. Ils sont rapides à prendre en main, leur exécution est optimisée et le rendu visuel est harmonieux. Les **customisations** possibles sont nombreuses : exploitation des événements Javascript des différents widgets, custom binding sur les grilles, clic sur une barre d'un graphique... mais il ne faut tout de même pas aller trop loin. Quand on a l'habitude d'utiliser une librairie, il arrive malheureusement qu'on ait tendance à vouloir utiliser ses composants à tout prix : on en a déjà ajouté sur d'autres pages, on veut quelque chose d'harmonieux visuellement, et cela peut donner l'illusion qu'avec ce composant, le développement de l'application ira plus vite. Comme c'est le cas pour tout framework, utiliser des composants pour utiliser des composants peut être sérieusement **contre productif**. Par exemple, la Grid Kendo est parfaite quand on dispose d'une liste de colonnes fixe. Utiliser une liste de colonnes dynamique n'est toutefois pas impossible.
Imaginons par exemple une liste de produits ; certaines de leurs propriétés, de types divers, seraient définies en backoffice par un administrateur, et disponibles sous la forme d'un Hashtable au sein de l'objet. Il serait alors possible d'accéder à toutes ces propriétés en itérant sur le Hashtable, et de les présenter sous forme de colonnes dans une Grid Kendo.

```
@(Html.Kendo().Grid<Programmez.Example.Models.MyViewModel>
    (Model)
    .Name("Grid")
    .Columns(columns =>
    {
        columns.Bound(o => o.Id).Width(50);
        // Liste des headers de colonne
        foreach (var col in (IEnumerable<Programmez.Example.Models.ColumnView
Model>)ViewData["columnsList"])
        {
            columns
                .Template(o => o.ProductFields.Keys.Contains(col.Id)
                    ?  o.ProductFields[col.Id].ToString()
                    :  string.Empty
                )
                .Title(col.ColumnName).Width(150);
        }

    })
)
```

(7) https://plugins.jquery.com/loadTemplate/
(8) https://mustache.github.io/
(9) http://handlebarsjs.com/
(10) https://fr.wikipedia.org/wiki/TypeScript
(11) https://fr.wikipedia.org/wiki/Scalable_Vector_Graphics
(12) https://fr.wikipedia.org/wiki/Canvas_(HTML)
(13) https://fr.wikipedia.org/wiki/Vector_Markup_Language
(14) http://www.igniteui.com/help/using-infragistics-loader
(15) https://en.wikipedia.org/wiki/Asynchronous_module_definition
(16) http://docs.telerik.com/kendo-ui/aspnet-mvc/how-to/use-with-requirejs-mvc
(17) http://docs.telerik.com/kendo-ui/third-party/systemjs

Mais les fonctionnalités avancées comme les tris, les filtres, le groupement par colonne, risquent d'être très compliquées à mettre en place et s'avérer gourmandes au niveau des ressources de la page, lorsqu'une solution "maison" plus simple se révèlerait sans doute plus efficace.

Quelques lacunes

Certaines fonctionnalités qu'on attendrait **nativement** sur les composants sont parfois absentes et totalement hors de la roadmap, malgré les demandes répétées des utilisateurs sur les forums. C'est le cas par exemple de l'enregistrement de l'état d'une Grid : la possibilité de grouper ou filtrer de manière très fine les données affichées, amène à s'attendre à retrouver la grille dans l'état dans lequel on l'a laissée si on navigue sur une autre page et qu'on revient par la suite sur la page. L'éditeur propose alors un bricolage pour **contourner** le problème : exploiter l'événement DataBound de la grille pour récupérer son état (filtres sélectionnés mais aussi page courante, groupements effectués, etc), le sauver localement dans la mémoire locale du navigateur ou en session via une requête AJAX, et le recharger de la même manière quand on revient ultérieurement sur la page.

```
@(Html.Kendo().Grid<Programmez.Example.Models.MyViewModel>()
    .Name("grid")
    .DataSource(ds => ds.Ajax().Read("Get", "Grid"))
    .Events(e => e.DataBound("grid_dataBound"))
    .Pageable()
    .Groupable()
    .Sortable()
)

<script type="text/javascript">

  // sauvegarde du contexte de la grille
  function grid_dataBound() {
    var dataSource = $("#grid").data("kendoGrid").dataSource;

    var state = {
      columns: grid.columns,
      page: dataSource.page(),
      pageSize: dataSource.pageSize(),
      sort: dataSource.sort(),
      filter: dataSource.filter(),
      group: dataSource.group()
    };

    $.ajax({
      url: '/Grid/SaveState',
      data: { 'data': JSON.stringify(state) },
      contentType: "application/json; charset=utf-8"
    });
  }

  // Chargement de la grille avec son contexte sauvegardé
  $(document).ready(function () {
    $.ajax({
      url: "/Grid/LoadState",
      success: function (state) {
        state = JSON.parse(state);

        var options = $("#grid").data("kendoGrid").options;

        options.columns = state.columns;
        options.dataSource.page = state.page;
        options.dataSource.pageSize = state.pageSize;
        options.dataSource.sort = state.sort;
        options.dataSource.filter = state.filter;
        options.dataSource.group = state.group;

        grid.destroy();

        $("#grid").empty().kendoGrid(options);
      }
    });
  });
</script>
```

Cela amène donc à charger la grille une première fois, puis à récupérer l'état sauvegardé de la grille à l'événement DOMReady pour la détruire et la recharger… C'est assez surprenant. Une solution consisterait à ne pas créer la grille au chargement et d'attendre de récupérer l'état sauvé, mais dans ce cas le wrapper ASP.NET devient inutile. De plus, les librairies très généralistes comme Kendo UI sont certes pratiques à mettre en place et offrent de nombreuses possibilités de composants, mais il est difficile d'imaginer que les fonctionnalités qui y sont disponibles égalent en variété celles des **librairies spécialisées** dans chaque domaine. Dans le cas de **DataViz** de Kendo UI par exemple, les librairies comme Highcharts(18), D3 (Data Driven Documents)(19), Fusioncharts(20), etc, proposent elles aussi des graphiques au format SVG, mais bien-sûr avec de nombreuses **fonctionnalités supplémentaires**. Selon l'utilisation qu'on souhaite faire de DataViz, il est parfois moins compliqué de passer directement par ce genre de librairies. Par exemple, dans le cas de graphes en colonnes complexes, Highcharts permet d'explorer une barre du graphique pour en voir le détail ("Drilldown")(21). Pour obtenir le même résultat avec Kendo UI, il faut ajouter soi-même la fonctionnalité en exploitant l'événement SeriesClick d'une barre pour modifier la source de données du graphe, ou en générer un deuxième, afin de présenter les données recherchées. Ces **intelligents workarounds** rendent de nouvelles choses possibles et permettent d'adapter la librairie à un cas précis d'utilisation de notre application, mais on perd assez rapidement le bénéfice d'une installation et d'une mise en place rapide du composant.

CONCLUSION

Les éditeurs proposent un **grand nombre de composants** pour agrémenter les applications, et dans certains cas, des wrappers serveurs qui représentent un gain de temps de développement conséquent. La **documentation** des librairies sous forme de Reference API, tutoriaux, designers et démos, est satisfaisante, et développer des interfaces modernes et présenter des données sous forme de tableaux, arbres ou graphes riches de fonctionnalités n'a jamais été aussi facile. Les widgets sont aisément **extensibles** pour s'adapter à tous les projets, et restent **performants** si toutefois on est raisonnable dans l'usage auquel on les destine. Les librairies de composants peuvent en conséquence être un **vrai plus** sur un projet Web ou mobile. •

(18) http://www.highcharts.com/
(19) https://d3js.org/
(20) http://www.fusioncharts.com/
(21) http://www.highcharts.com/demo/column-drilldown

Programmez sur **Atari ST** 30 ans après

Frédéric Sagez
Codeur du groupe
NoExtra-Team
fsagez@gmail.com

Fan de lignes de Punchs écrites en GFA-BASIC ou de codes assembleurs prodigués par un Jedi pour faire de beaux effets ? Vous avez eu la fièvre de la programmation sur votre ST lors de votre adolescence, il est toujours temps de s'y remettre. « Chérie, tu as mis où les ST Mag ? ». Euh…

Mais avant d'en arriver là, replongeons-nous quelques instants dans le code assembleur tout en étant aidé par un Framework, regardez comment est agencé un programme, faire tourner des effets, juste histoire de s'y remettre un peu plus facilement…

Et pour commencer...

Ce dont nous avons besoin dans le cas idéal : un Atari STF avec 512 ko ou 1 Méga de mémoire vive avec un écran couleur et une disquette contenant le programme DevPac ST pour compiler nos programmes.
A défaut d'être équipé du vrai hardware, il vous faudrait pour votre machine actuelle utiliser un émulateur tel que **Steem SSE** (https://sourceforge.net/projects/steemsse/) ou **HATARI** (https://hatari.tuxfamily.org/). La dernière version de Steem SSE permet d'émuler un Atari STF ou STE avec pas mal d'options avec une meilleure gestion du Shifter – partie graphique du ST – qui permet notamment de visualiser des démos ou des jeux utilisant le mode OVERSCAN du Hardware. Cet émulateur fonctionne uniquement sur Windows et peut utiliser des plugins. HATARI est un émulateur qui a fait du chemin, et qui, aujourd'hui, est arrivé à maturité ; fonctionnant sur MAC et PC, il offre aussi une foule d'options concernant la partie graphique est sonore plus la gestion de disque dur. Point important avant de commencer : la version du TOS (Operating System de l'Atari) est primordiale suivant que vous allez coder sur un Atari STF ou STE. Récupérez sur Internet une version 1.4FR pour écrire des programmes fonctionnant sur un Atari STF et une version 1.62FR spécifiquement sur STE. Il existe d'autres versions du TOS en cours de développement comme EmuTOS (http://emutos.sourceforge.net/) qui est un clone Open Source utilisé par défaut dans l'émulateur HATARI.

L'assembleur ? Kezako ?

L'assembleur est un langage de programmation constitué de jeux d'instructions représentés par des labels, mnémoniques, opérandes et commentaires qui sera compilé / traduit en langage machine pour être exécuté sur la machine cible. Pour écrire un programme en assembleur, vous avez besoin de connaître les différents compartiments ou mettre le code et les éléments pour que le programme se compile. Nous avons trois sections apparentes à indiquer : TEXT, DATA et BSS. La partie TEXT contient le code en assembleur. Elle est située au début du fichier. La partie DATA contient toutes les données « non modifiables » du programme (image, musique, etc.). La partie BSS sera la partie mémoire d'accès direct en lecture/écriture. Pour créer un exécutable et l'exécuter sur le bureau GEM, nous aurons besoin d'un compilateur, outil qui va convertir notre programme (appelé code source) en un exécutable binaire. Les codes sources utilisés sont des fichiers avec l'extension « .S ».
Dans notre cas nous allons utiliser le logiciel DEVPAC ST qui sert à la fois d'éditeur de texte, à compiler notre fichier en programme binaire ou à l'exécuter / déboguer dans une partie de la mémoire vive de la machine. Récupérer le répertoire DEVPAC et son contenu dans le repository et copiez-le à la racine du lecteur C de votre émulateur préféré. Pour l'exécuter, il faudra lancer le programme *DEVPAC.PRG* pour l'utiliser.

Remarque : il existe bien sûr d'autres compilateurs 68k que vous pouvez aussi utiliser sur différentes plateformes tel que VASM.

A propos du framework

Nous avons besoin d'avoir un programme qui puisse s'exécuter normalement sous GEM - le bureau de l'Atari - et le Framework que nous allons utiliser permet de répondre à différents critères.
Son mode de fonctionnement est assez simple d'utilisation et est agencé comme ceci :

- Il réserve de la mémoire et l'alloue automatiquement tout en adaptant tout le contenu du programme qui sera copié au chargement et exécuté en mémoire lors de son lancement,
- Nous utilisons le mode SUPERVISEUR pour utiliser certaines fonctions du TOS et accéder à des zones de mémoires spécifiques du ST sans provoquer de plantage machine avec des bombes à l'affichage.
  Nous allons très peu utiliser d'appel système jusqu'à la fin du programme. (Appel de type TRAP #1 soit un appel GEMDOS)
- Nous préparons la stack BSS dans un premier temps, puis nous allons initialiser le Hardware du ST : nous sauvegardons l'état des Timers et des Interruptions du ST tout en les désactivant afin de démarrer notre pro-

gramme « proprement », on sauvegarde la palette d'origine - soit seize couleurs - puis nous initialisons les états des Timers ainsi que les Interruptions au gré de nos besoins. Et enfin nous sauvegardons l'état des adresses vidéo d'origine, on supprime la souris, le son du clavier et on prépare l'ACIA pour gérer les flux entrants / sortants du clavier.

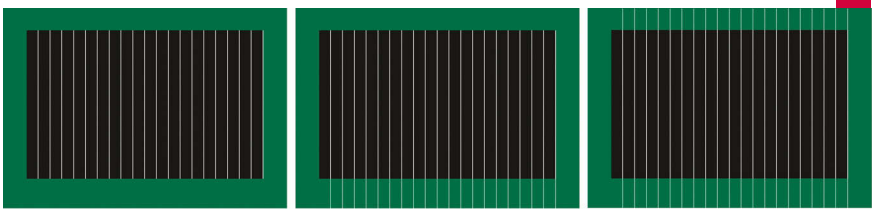
- Côté graphique, nous allons préparer deux écrans « physiques » et « logiques » que nous allons créer dans notre zone BBS en full accès mémoire afin de permettre un clipping parfait à chaque fois que le ou les effets seront exécutés dans la boucle principale *default_loop,* tout en étant synchronisés avec la **VBL**. Nous pouvons appliquer un masque à nos écrans avec un *PATTERN* pour avoir une visualisation complète des écrans utilisés, mais il permet d'abord d'effacer proprement nos écrans. A noter que parfois nous n'avons pas forcément besoin de deux écrans, suivant les accès mémoire et le plan - bitplane - utilisé par le ou les effets. C'est pourquoi nous pouvons choisir le nombre d'écrans à utiliser, soit un ou deux par défaut. Mais il ne faut pas négliger la mémoire utilisée au final car un écran représente une zone mémoire de 160 x 200, soit 32kb.
- Nous initialisons enfin la musique, puis nous lançons l'exécution de la VBL qui va nous permettre de synchroniser tous les évènements de notre programme : jouer de la musique au format Soundchip en boucle et de gérer les différents Timers côté **MFP**.
- Puis nous avons une boucle principale qui permet d'échanger nos deux écrans physiques et logiques tout en incluant des effets que l'on pourra exécuter à l'intérieur de celle-ci. On pourra sortir de la boucle en appuyant la touche ESPACE ou on pourra visualiser le temps machine restant via la touche ALTERNATE de gauche du clavier.
- Et au final nous rendons l'état original au système tout en passant le mode standard du ST, le mode USER.

Description du Framework

Nous allons utiliser des directives d'assembleur comme paramètres pour paramétrer notre environnement de démonstration :

- Les paramètres *BOTTOM_BORDER*, *TOPBOTTOM_BORDER* permettent de passer en mode Overscan bas ou haut plus bas tout en jouant avec les Timers du ST ou en étant en mode standard avec l'option *NO_BORDER*. A la base, l'écran du ST aura une résolution 320 par 200 avec 16 couleurs, soit le mode LOW RESOLUTION à 50 Hertz. Pour paramétrer le type d'écran voulu, il suffit de choisir son mode d'affichage en mettant à "0" celui que l'on désire utiliser et de mettre à "1" les deux autres dont on n'a pas besoin. [1]

1



NO_BORDER BOTTOM_BORDER TOPBOTTOM_BORDER

La zone de couleur verte correspond à la zone non modifiable de l'écran d'un ST.

- Concernant la gestion des écrans, nous utiliserons deux écrans par défaut (physiques et logiques) pour éviter des effets de scintillement via le paramètre *NB_OF_SCREEN*. Il doit être initialisé avec le paramètre *ONE_SCREEN* si on utilise un écran ou *TWO_SCREENS* pour deux écrans. Le paramètre *PATTERN* est notre masque pour remplir nos écrans, on peut mettre des valeurs à partir de $0 jusqu'à $ffffffff. Mais il doit toujours être initialisé car il sert principalement à les effacer.
- Le paramètre *SEEMYVBL* permet de visualiser le temps machine qui est utilisé par le ou les effets.
- Le paramètre *ERROR_SYS* capture le type d'erreur. Au lieu d'afficher un nombre de bombes accompagné d'un reset Hard, on indique via un code couleur où se situe notre erreur. (Bus de données, Bus d'adresse, etc.).

La gestion des erreurs a été écrite par Dbug (Mickaël Pointier - http://www.defence-force.com)

- Le paramètre *FADE_INTRO* permet d'effectuer un Fade de couleurs à partir de la couleur blanche provenant du GEM vers la couleur noire.
- Le paramètre *TEST_STE* détecte si le type d'ordinateur utilisé est un STE.
- Le paramètre *STF_INITS* utilisé par défaut, permet au programme de rester compatible avec les autres gammes Atari tel que le ST jusqu'au FALCON avec carte accélératrice lors de son exécution. Source écrit par Leonard (Arnaud Carré - http://leonard.oxg.free.fr/SainT/saint.html).

Grâce à ce petit programme écrit en assembleur nous allons écrire notre première démonstration qui sera constituée des effets de RASTERS et d'un SCROLLTEXT.

Dépassons les capacités de l'Atari STF

En utilisant le Timer B, nous allons changer la couleur du fond de l'écran, et ce, permettre d'afficher plus de 16 couleurs ! Nous pouvons par défaut utiliser 199 lignes de couleurs sur la partie de l'écran - visible - standard du ST, et pour chaque ligne nous pouvons afficher une couleur qui s'étalonne entre $000 et $777. Pour cela nous allons utiliser les Timers A et B via les interruptions $fffffa07.w et $fffffa13.w. Nous allons sélectionner à partir de quelles lignes on va afficher nos Rasters en passant par l'adresse $fffffa21.w, initialiser notre routine **HBL** en la plaçant sur l'adresse du vecteur $120.w, puis lancer notre Timer via l'interruption $fffffa1b.w. Notre routine exécutée dans la **HBL** permet de temporiser / synchroniser le Timer B et l'adresse vidéo du ST (adresse $ffff8209.w) pour ensuite envoyer notre palette de couleur que l'on veut afficher via l'adresse de la couleur de notre fond d'écran qui est à l'adresse $ffff8240.w. (40 couleurs au total vont être utilisées pour cette palette non standard).

Une fois la synchronisation faite et les variables implémentées, nous utilisons une boucle pour afficher 160 lignes de Rasters en forme d'escalier. Nous envoyons les couleurs de notre palette *PAL_RAST* via le registre d'adresse A0 sur notre couleur de Background sur le registre d'adresse A1 via l'instruction MOVE.W qui transmet des données au format WORD. Nous obtenons un effet improbable qui est en fait « customisé » par un nombre de NOP (instruction n'effectuant rien) bien « placé » tout en jouant avec les Timers !

Voici le code utilisé pour cet effet, ne pas oublier de protéger les registres utilisés dans notre **HBL** afin de ne pas impacter tout la démonstration : [2]

Petit rappel:

- Toutes les instructions en dehors des Overscans sont incluses dans la directives *NO_BORDER* (initialisation et lancement) ;
- La Vertical Blank Line utilise le vecteur d'interruption $70.w, c'est elle qui lance /gère des Timers en attendant le retour de l'écran, on la synchronise à chaque balayage dans la

```
Hbl_ligne:
    clr.b     $fffffa1b.l

    movem.l   a0-a1/d0-d1,-(a7)

    moveq     #$e,d0
    move.w    #9,d1
.loop:
    dbf       d1,.loop              ; Tempo !
.sync:
    move.b    $ffff8209.w,d1        ; We have a video adress available ?
    beq.s     .sync
    sub.b     d1,d0
    rol.w     d0,d0                 ; Set done !
    lea       $ffff8240.w,a1        ; Color of the Backgound
    lea       PAL_RAST,a0           ; Colors of the Rasters
    move.w    #9,d0
.tempo:
    dbf       d0,.tempo             ; Tempo !

    move.w    #80,d0                ; Rasters Loop
.DoRaster:
    dcb.w     40,$3298              ; 40 instructions MOVE.W (A0)+,(A1)
    dcb.w     11,$4e71              ; 11 NOP
    dcb.w     40,$3298              ; 40 instructions MOVE.W (A0)+,(A1)
    dcb.w     8,$4e71               ; 8 NOP
    dbf       d0,.DoRaster

    movem.l   (a7)+,a0-a1/d0-d1

    bclr      #0,$fffffa0f.w        ; Stop Timer A & B
    move.b    #0,$ffff8240.w        ; Black color after Rasters
    rte
```

2

```
Scrolling:
    move.l    physique(pc),a0       ; Physique screen selected for display the effect
    lea       160*100(a0),a0        ; And put the Scrolltext in the middle of the screen
    lea       Buffer_Scroll,a2      ; Buffer of the character font
    moveq     #15,d7                ; 16 lines to display
.scroll:
    roxl      (a2)+                 ; Scroll each pixel of the character Buffer 16 times
i set 160-8
 rept 20
    roxl      i(a0)                 ; Scroll the right to the left part of the Screen
i set i-8
 endr
    lea 160(a0),a0                  ; Next line
    dbf d7,.scroll

    addq      #1,sequence           ; We need to ROXL 16 pixels of the character
    cmpi      #16-1,sequence        ; And we need to count 16 times
    ble       nocar
    clr       sequence
restart:
    move.l    pt_text,a0            ; Next character of the text
    move.b    (a0)+,d0
    tst.b     d0
    bne.s     .noendtxt
    move.l    #text,pt_text         ; Or restart the text
    bra.s     restart
.noendtxt:

    move.l    a0,pt_text            ; Test character in our ASCII table
    lea       list_chr,a2
    moveq     #-$1,d1
.search:
    addq      #$1,d1
    cmp.b     (a2)+,d0
    bne.s     .search
    lea       fonte,a0              ; Find Font character to display
search0:
    cmpi      #20,d1
    bmi.s     .search1
    lea       640(a0),a0
    subi      #20,d1
    bra.s     search0
.search1:
    add       d1,d1
    lea       0(a0,d1.w),a0         ; Find it !

    lea Buffer_Scroll,a2            ; Recopy Font character to the Buffer character
    moveq     #16-1,d0
.recopy:
    move      (a0),(a2)+
    lea       40(a0),a0
    dbf       d0,.recopy
nocar:
    rts
```

3

boucle principale avec la fonction *Wait_Vbl* ;

- La Horizontal Blank Line balaye l'écran ligne par ligne et utilise le vecteur d'interruption $120.w ;
- Ses deux interruptions matérielles sont reconnaissables car elles finissent par un RTE (Return from Exception) ;
- La Multi Fonctional Peripheral - appelé **MFP** - gère les interruptions matérielles que nous avons initialisées juste avant d'installer les deux vecteurs d'interruption.

Le scrolltext

Pour faire un scrolltext nous allons utiliser la résolution standard du ST qui est de 320 pixels par 200 lignes. Chaque pixel est composé d'une valeur entre 0 to 15 (1 pixel est égale à 4 bits), soit 160 bytes affichées par ligne ? Prenons exemple avec la valeur $FFFF0000, notre byte, notre pixel donc équivaut à la couleur numéro 6 de notre palette qui se situe au 3ème plan sur quatre que peut utiliser notre écran. Pour faire bouger un élément sur notre écran physique du ST, même en utilisant le 1er plan qui contient une seule couleur, nous devons modifier en temps réel le ou les pixels, ce qui est très consommateur pour une machine architecturée à base d'un Motorola 68000 à 8Mhz. Pour pallier ce problème, nous devons calculer les décalages requis pour chaque déplacement du bloc de pixels via l'instruction Rotate Left with Extend !

Donc pour afficher un Scrolltext de 16 par 16 pixels, soit une ligne continue de 320 pixels sur 16 lignes, nous n'allons pas utiliser un scrolling standard avec l'instruction MOVE.W mais plutôt l'instruction ROXL.W qui permettra de gérer le déplacement à gauche de la partie graphique tout en utilisant le registre étendu "X" à chaque décalage. De plus nous utiliserons un Buffer pour chaque caractère à afficher, en fait cela nous permettra de bufferiser notre séquence dans une partie de la mémoire et de la copier ensuite sur l'écran ; ceci afin d'éviter de tout le temps solliciter l'adresse vidéo pour chaque instruction effectuée lors du Scrolling. Nous écrirons un sous-programme *Scrolling* que nous appellerons via un BSR dans la boucle principale.

Voici le code du scrolling de 1 pixel par frame : [3]

Avec le code des Rasters dans la VBL et l'appel du Scrolling dans la boucle principale, voici le résultat final haut en couleur : [4]

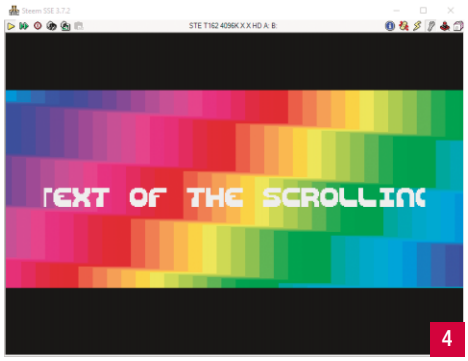
Vous trouverez le code source assembleur complet sur la branche suivante : https://github.com/NoExtra-Team/framework/tree/master/sources/RASTERS.

Ressources

Le Framework est disponible à l'adresse suivante https://github.com/NoExtra-Team/framework et rien ne vous empêche de contribuer à ce petit bout de code pour le faire évoluer ou de créer d'autres exemples. Pour plus d'informations lisez le fichier A_LIRE.TXT. Le programme DevPac ST de l'éditeur HiSoft avec lequel on va compiler nos fichiers écris en assembleur est disponible sur une autre branche https://github.com/NoExtra-Team/framework/tree/master/utils. La documentation de DevPac est disponible au format PDF sur le site Internet http://devlynx.ti-fr.com/ST/, vous pourrez ainsi connaître tous les secrets de la compilation sur ST avec ce logiciel.

Documentations

- Les livres du développeur TOME 1 et 2 pour tout comprendre sur la technologie des Atari modèles STF et STE



4

- Une foule de scans de livres concernant l'Atari ST sont hébergés sur le site http://devlynx.ti-fr.com/ST/
- Plus de codes sources et de tutos ? Visitez le Wiki *Atari-forum* sur http://www.atari-wiki.com/?title=Assembly_language_tutorials
- Tous les utilitaires pour Atari ST sont disponibles sur le site *Essential software List* à l'adresse https://sites.google.com/site/stessential/
- Fan du magazine *ST Magazine*, tous les disques sont archivés et disponibles sur http://stmagazine.free.fr/archives/

Pour finir

Nous verrons dans un prochain article la gestion d'un menu avec le chargement et l'exécution de programme, en fait nous verrons comment sont agencés les Menus EXTRA par le groupe de démo NoExtra-Team. Nous parlerons aussi du Blitter sur STE et comment jouer un Module quatre voix sur un Atari STF et sur STE. Bref, tout plein de bonnes choses pour la prochaine fois....

RxJS pour les humains

Nicolas Baptiste
Concepteur Développeur
SQLI

Bien malin est celui qui aurait pu prédire l'avenir du Javascript à sa création ! Parent pauvre de la programmation, considéré comme un langage de bidouilleur, il a désormais entièrement sa place dans les front-ends les plus complexes mais aussi côté backend avec NodeJS.

Le langage a évolué ainsi que les outils permettant de le manipuler (frameworks, IDE, outils de débogage, etc.). Et grâce à Angular ou encore Cycle.js on entend de plus en plus parler de RxJS et de programmation réactive. Quelques mots barbares pour nous autres pauvres développeurs obligés il y a encore quelques temps de placer des 'alert' pour déboguer nos programmes. Avant de rentrer directement dans le vif du sujet, remontons un peu le temps pour comprendre comment nous en sommes arrivés là et surtout à quelle problématique répond la programmation réactive. Prenons un bout de code tout simple :

```
var foo = 'a';

if (foo === 'a') {
 console.log('foo = a');
} else {
 console.log('foo != a');
}

alert('Voilà tout est fini');
```

On commence tout doucement. Le code s'exécute ligne par ligne, chaque instruction est bloquante et si pour une obscure raison l'exécution de la fonction `console.log` prenait 3 secondes, l'instruction `alert` s'exécuterait après 3 secondes. Jusque-là tout va bien. Et puis un beau jour vint le code asynchrone avec des termes comme xhr et AJAX.

```
var req = new XMLHttpRequest();
req.open('GET', 'http://www.mozilla.org/', true);
req.onreadystatechange = function (aEvt) {
  if (req.readyState == 4) {
     if(req.status == 200)
      dump(req.responseText);
     else
      dump("Erreur pendant le chargement de la page.\n");
  }
};
req.send(null);
```

Source: https://developer.mozilla.org/fr/docs/Web/API/XMLHttpRequest

On instancie un objet de type XMLHttpRequest qui permet de requêter une adresse. Puis à l'aide d'une astucieuse fonction enregistrée via `onreadystatechange` notre bout de code est exécuté lorsque le réseau daigne nous répondre.
Ce n'était déjà pas mal, mais la syntaxe n'est pas très jolie. En plus la version d'Internet Explorer de l'époque ne proposait qu'un `ActiveXObject` et il fallait en tenir compte car c'était de loin le navigateur le plus utilisé.
Puis il y eu la révolution jQuery ! Ah, le sentiment de puissance et de facilité avec `$.ajax` ou même `$.get` ! Toute la difficulté d'instanciation de nos requêtes asynchrones était masquée dans une API relativement simple : http://api.jquery.com/jquery.ajax/

```
$.get( "ajax/test.html", function( data ) {
  $( ".result" ).html( data );
  alert( "Load was performed." );
});
```

Source: http://api.jquery.com/jQuery.get/

En 4 lignes de code c'est plié ! Un appel, puis une fonction de rappel (callback en anglais) est exécutée si l'appel aboutit, une autre si l'appel échoue.
Mais très vite nous sommes tombés sur ce que l'on appelle : *LE CALLBACK HELL* ! Un nom qui fait très peur pour décrire une fonction de rappel faisant un autre appel asynchrone qui déclarait lui-même un autre rappel qui faisait lui-même un autre appel asynchrone et ainsi de suite. Regardons cet exemple en NodeJS :

```
fs.readdir(source, function (err, files) {
  if (err) {
    console.log('Error finding files: ' + err)
  } else {
    files.forEach(function (filename, fileIndex) {
      console.log(filename)
      gm(source + filename).size(function (err, values) {
        if (err) {
          console.log('Error identifying file size: ' + err)
        } else {
          console.log(filename + ' : ' + values)
          aspect = (values.width / values.height)
          widths.forEach(function (width, widthIndex) {
            height = Math.round(width / aspect)
            console.log('resizing ' + filename + 'to ' + height + 'x' + height)
            this.resize(width, height).write(dest + 'w' + width + '_' + filename, function(err) {
              if (err) console.log('Error writing file: ' + err)
            })
          }.bind(this))
        }
      })
    })
  }
})
```

Source: http://callbackhell.com/

Je n'ai même pas envie de décrire ce bout de code. Il est horrible à lire et encore plus dur à déboguer en cas de problème. Mais que celui ou celle qui n'a jamais codé de la sorte jette la première pierre !
Mais un (autre) beau jour, une solution élégante à ce problème est appa-

rue : les promesses. On peut citer notamment la librairie Q de Kris Kowal. Il le dit lui-même, sa librairie peut gérer ce qu'il appelle *the Pyramid of Doom*.

```
step1(function (value1) {
  step2(value1, function(value2) {
    step3(value2, function(value3) {
      step4(value3, function(value4) {
        // Do something with value4
      });
    });
  });
});
```

devient:

```
Q.fcall(promisedStep1)
  .then(promisedStep2)
  .then(promisedStep3)
  .then(promisedStep4)
  .then(function (value4) {
    // Do something with value4
  })
  .catch(function (error) {
    // Handle any error from all above steps
  })
  .done();
```

Le paradigme des promesses est le suivant, par exemple : j'envoie ma petite sœur chercher du pain, puis quand elle reviendra je le découperai puis je me beurrerai une tartine. Je prévois toute ces actions à l'avance mais je ne pourrai les effectuer qu'à son retour. En attendant je vais aller faire un peu de jardinage !
Si on code ça dans AngularJS par exemple (qui implémente sa version des promesses via le service $q), cela nous donne :

```
function sendLittleSisterToFetchTheBread () {
  var deferred = $q.defer();
  var plombes = 1000;

  setTimeout(function () {
    deferred.resolve('bread');
  }, 3 * plombes);

  return deferred.promise;
}

sendLittleSisterToFetchTheBread()
  .then(function (bread) {
    console.log('I cut the ' + bread);

    return 'toast';
  })
  .then(function (toast) {
    console.log('I use the ' + toast + ' to put my butter on it');
  });

workInTheGarden();
```

Ainsi, toute la logique permettant à ma petite sœur d'aller chercher le pain, puis ensuite d'utiliser le pain est mise en place de manière élégante et lisible. Je n'ai donc pas à attendre 3 plombes pour travailler dans le jardin. Avec les promesses on résout beaucoup de cas fonctionnels en produisant un code assez propre.
Mais forcément, on en veut toujours plus. Par exemple : comment dire à ma petite sœur, qu'en fait, du pain il y en avait déjà dans le congélateur ? La petite n'ayant pas de téléphone portable, impossible de lui dire d'arrêter sa course. Et oui, une promesse ne peut pas être annulée.
Ou encore, une promesse résolue ne peut pas être réexécutée si elle a échoué... Une promesse effectue une action asynchrone, mais comment faire pour réagir à plusieurs événements asynchrones ?

La programmation réactive à notre secours

Aujourd'hui lorsqu'on parle d'asynchrone, on pense très vite à un appel HTTP vers une ressource externe (le traditionnel AJAX). Mais il s'agit simplement de réagir à un événement qui se produira à un moment donné. De la même manière nous pouvons réagir à tout autre événement sans savoir quand il se produira (click, input, websocket, etc.).
Maintenant imaginez un code qui calcule le nombre de clics d'un utilisateur : facile, une variable initialisée à 0 qui s'incrémente à chaque callback enregistré sur l'événement `onclick`.
Ensuite imaginez un code qui calcule le temps en millisecondes entre deux clics et qui se met à jour en fonction du dernier clic. *On fait moins les malins là.*
On imagine deux variables dont on calculerait la différence de timestamp, puis qu'on intervertirait dès qu'un nouveau clic arrive... C'est faisable, mais le résultat ne serait pas très beau.
Et là RxJS vient à notre rescousse. Cette librairie va nous permettre de traiter l'ensemble de ces événements comme un flux (en anglais *stream*) qui s'apparentera ni plus ni moins à un tableau.
Et cerise sur le pompon, elle nous offrira tout un tas de méthodes pour traiter ce flux, le filtrer, le transformer ou même le combiner avec d'autres flux !

```
Rx.Observable.fromEvent(document, 'click')
        .map(() => Date.now())
        .pairwise()
        .map(([before, after]) => (after - before))
        .subscribe(x => console.log(x));
```

Pour chaque clic, on retourne 1 timestamp, puis on groupe chaque timestamp par 2 avec `pairwise`. Une autre fonction `map` (destructurée via la syntaxe ES6) calcule la différence entre les 2 timestamps et enfin via `subscribe`, on redirige la sortie vers la console.
Et voilà ! Avec RxJS et ES6 on résout ce problème en 5 lignes de code parfaitement claires.

Penser en termes de flux / stream

Cette partie est une traduction de l'excellent article '*The introduction to Reactive Programming you've been missing*' https://gist.github.com/staltz/868e7e9bc2a7b8c1f754 par Andre Medeiros et http://andre.staltz.com
Un flux est une séquence d'événements en cours, ordonnés dans le temps. Un flux peut émettre 3 choses : des valeurs, des erreurs et un signal de complétion indiquant que le flux est terminé et ne renverra plus de valeurs.

```
--a---b-c---X---d--- | ->

a, b, c, d sont des valeurs
X est une erreur
| représente signal 'completed'
---> représente le temps
```

Avec RxJS vous pouvez appliquer des fonctions sur un flux et produire d'autres flux.
Par exemple avec la méthode map (utilisée dans l'exemple plus haut).

```
clickStream: ---c----c--c----c------c---------->
                vvvv map(x = > Date.now()) vvvv
                ---t1---t2-t3---t4-----t5--------->

t*  == timestamp courrant
```

RxJS est une implémentation du design pattern Observer : https://fr.wikipedia.org/wiki/Observateur_(patron_de_conception) Les flux sont observables et nous réagissons via des fonctions observers.

Chaud et froid

Certains observables ne produiront aucune valeur s'ils ne sont pas écoutés ou observés via la fonction subscribe. On les qualifie de *froids* (*cold* en anglais).

```
const arrayObservable = Rx.Observable
      .fromArray([1, 2, 3, 4])
      .map(x => x * 10)
      .tap(x => console.log(x));

// rien ne s'est encore produit
arrayObservable.subscribe();

// affiche en console
// 10
// 20
// 30
// 40
```

A l'inverse certains sont qualifiés de *chauds* lorsque des valeurs sont produites même si le flux n'a pas de souscription.
C'est le cas lorsque l'on crée un flux pour écouter les clics de l'utilisateur. Des valeurs sont produites même si l'on ne souscrit pas à cet observable (ça semble logique).
En revanche, les observables sont *fainéants* (*lazy* en anglais) et ne seront traités, chauds ou froids qu'à l'appel de la fonction subscribe().
Une dernière chose à savoir, est que la fonction subscribe renvoie un objet de type Disposable qui permettra d'arrêter l'écoute d'un flux. Et donc l'arrêt de production de valeurs s'il s'agit d'un observable froid.

Et après ?

Vous commencez à comprendre l'intérêt de RxJS ? Mais comment aller plus loin ? Regardons d'abord comment créer des observables. La librairie nous offre une multitude de méthodes permettant de créer des observables à partir de tableaux, de promesses, d'événements, de callbacks... Et la liste pourrait encore continuer.

Ce qu'il faut ensuite comprendre c'est que l'on résonne désormais en termes de flux et qu'on les manipule pour en obtenir une sortie.
Un très bon site existe pour comprendre l'usage des fonctions de RxJS est http://rxmarbles.com/
Il représente les flux de cette manière :

- Une flèche de gauche à droite représente le temps ;
- Des ronds sont des valeurs constituant le flux et renvoyés à un moment précis ;
- Une barre sur la flèche du temps indique lorsque le flux ne renvoie plus de données et que son statut est "complete" ;
- Entre 2 flèches de temps, la fonction utilisée pour transformer le premier flux en un autre flux.

Voyons par l'exemple. Vous connaissez peut-être la fonction reduce qui permet d'accumuler des valeurs de tableau. Avec RxJS, la fonction reduce renverra l'accumulation des valeurs une fois que le flux aura envoyé l'évènement completed.
http://rxmarbles.com/#reduce
Mais si nous voulons produire un nouveau flux B pour chaque valeur que le flux A renvoie, il faudra alors utiliser la fonction scan.
http://rxmarbles.com/#scan

Non désolé je ne suis pas prêt...

Que vous le vouliez ou non le langage JavaScript évolue via les normes ECMAScripts. Les promesses sont natives en ES6/ES2015 et les observables sont proposés pour ES2016 ! Plus d'excuses, autant s'y mettre dès maintenant sinon dans peu de temps vous serez complètement largués !
https://github.com/tc39/ecma262

Mais dans la vraie vie je m'en sers comment ?

Pour répondre à cette question, allez chercher des ressources sur Internet, il y en a plein ! Vous pouvez commencer par les exemples de la librairie : https://github.com/Reactive-Extensions/RxJS/tree/master/examples
Ou encore, allez voir par ici pour savoir comment intégrer RxJS avec AngularJS ou JQuery : https://github.com/Reactive-Extensions/RxJS/tree/master/doc/howdoi
Pour lire la documentation de manière plus simple :
http://xgrommx.github.io/rx-book/index.html
Une petite précision assez importante, vous trouverez 2 versions majeures actuellement de la librairie RxJS : la 4 et la 5. Et pour compliquer le tout, elles ne sont pas sur le même répertoire Github !
La version 4, est la version stable actuelle https://github.com/Reactive-Extensions/RxJS
La version 5 est en beta https://github.com/ReactiveX/rxjs et est une réécriture de la version 4 qui est censée être plus stable, plus rapide mais attention l'API introduit des changements non rétro-compatibles avec la version 4.
RxJS est le lodash pour les événements
Ben Lesh (https://twitter.com/BenLesh) project leader de la version 5.
Si vous connaissez lodash, vous savez la richesse de cette librairie et le temps qu'il faut pour connaître l'ensemble de ses fonctions. En revanche, il n'est pas nécessaire de tout maîtriser pour commencer à utiliser lodash et il en va de même avec RxJS.
D'autant plus que les Reactive-Extensions peuvent être utilisées dans un tas de langages : .NET, C++, JS, Ruby et Python pour la version 4. Et encore plus pour la version 5 : PHP, Java, Scala, Go, et bien d'autres (https://github.com/ReactiveX) !
Les langages changent mais la logique reste la même. Alors qu'attendez-vous pour commencer ? •

Desktop App Converter : UWP pour tous !

• Jonathan ANTOINE
et Maxime FRAPPAT
Développeurs chez **Infinite Square**
Blog : http://blogs.infinitesquare.com/

Le projet Centennial que Microsoft couvait depuis un petit moment a débouché sur un nouveau produit : le Desktop App Converter. Ce produit est un convertisseur d'application de bureau (WPF, Windows Forms, Win32) pour aboutir à une application UWP. La conversion permet donc de générer un package UWP sous la forme d'un fichier (.appx) ou d'un package (.appxbundle) à partir de l'installateur de vos anciennes applications.

Bien entendu, les avantages à la clé sont nombreux : publication dans le Windows Store, accès aux API UWP, utilisation des tuiles (dynamiques ou non), utilisation des tâches en arrière-plan (background tasks). Comme vous vous en doutez, il y a bien sûr quelques prérequis pour que la conversion se passe correctement. Voici quelques spécifications obligatoires qui vous amèneront à adapter votre code :

- L'utilisation de .NET 4.6.1 est obligatoire ;
- Ne pas exiger de toujours lancer l'application avec des privilèges élevés ;
- Aucune modification de la clé HKEY_LOCAL_MACHINE n'est autorisée ;
- Ne pas utiliser le dossier AppData pour partager des données avec une autre application.

Convertir son application automatiquement

Pour récupérer le convertisseur, rendez-vous à l'adresse suivante : https://www.microsoft.com/en-us/download/details.aspx?id=52757. Une fois téléchargé, décompressez-le dans le répertoire de votre choix et vous y trouverez le script PowerShell DesktopAppConverter.ps1 qui sera notre point d'entrée. Ici, pas d'interface utilisateur, tout se fait en lignes de commandes J
Pour commencer, il faut installer la fonctionnalité 'Container' à partir d'une image (BaseImage-1xxxx.wim) disponible à la même adresse de téléchargement que le convertisseur. Ouvrez une fenêtre PowerShell en mode administrateur et tapez les commandes :

```
> Set-ExecutionPolicy bypass
> .\DesktopAppConverter.ps1 -Setup -BaseImage .\BaseImage-1XXXX.wim -Verbose
```

N'oubliez pas de remplacer le nom de l'image par celle que vous avez téléchargée auparavant (BaseImage-14361.wim par exemple). Un redémarrage sera peut-être nécessaire afin de finaliser l'installation.
Pour convertir une application, rien de plus simple :

```
> .\DesktopAppConverter.ps1 -Installer D:\putty.exe -InstallerArguments "/S"
Destination D:\PuttyUWP -PackageName "Putty" -Publisher "CN=InfiniteSquare"
-Version 1.0.0.0 -MakeAppx -Verbose
```

Et vous voilà avec un votre nouvelle application UWP.

Convertir son application manuellement

Si vous ne souhaitez pas utiliser l'outil de conversion automatique, il existe une méthode pour le faire manuellement afin de contrôler plus précisément ce qui est fait. Cela se fait en trois étapes simples.

Création du manifeste de l'application

Comme dans n'importe quelle application UWP, vous aurez besoin de créer un fichier nommé appxmanifest.xml contenant les informations de votre application. Ce manifeste doit contenir au minimum les données suivantes :

```
<?xml version="1.0" encoding="utf-8"?>
<Package
  xmlns="http://schemas.microsoft.com/appx/manifest/foundation/windows10"
  xmlns:uap="http://schemas.microsoft.com/appx/manifest/uap/windows10"
xmlns:rescap="http://schemas.microsoft.com/appx/manifest/foundation/windows
10/restrictedcapabilities">
 <Identity Name="PuttyUWP"
  ProcessorArchitecture="x64"
  Publisher="CN=Putty Corp"
  Version="1.0.0.0" />
 <Properties>
  <DisplayName>Putty UWP</DisplayName>
  <PublisherDisplayName>Putty Corp</PublisherDisplayName>
  <Description>Best app EVER</Description>
  <Logo>logo.png</Logo>
 </Properties>
 <Resources>
  <Resource Language="fr-fr" />
 </Resources>
 <Dependencies>
  <TargetDeviceFamily Name="Windows.Desktop" MinVersion="10.0.14316.0" Max
VersionTested="10.0.14316.0" />
 </Dependencies>
 <Capabilities>
  <rescap:Capability Name="runFullTrust"/>
 </Capabilities>
 <Applications>
  <Application Id="App" Executable="Putty.exe" EntryPoint="Windows.FullTrust
Application">
   <uap:VisualElements
    BackgroundColor="#BBBBBB"
    DisplayName="PuttyUWP"
    Square150x150Logo="logo_150.png"
    Square44x44Logo=" logo_44.png "
    Description="Best app EVER" />
```

```
  </Application>
 </Applications>
</Package>
```

Création du package

Un fois le manifeste créé, il ne reste plus qu'à générer le package grâce à l'exécutable MakeAppx.exe (inclus dans le SDK Windows 10). Il existe deux méthodes pour gérer le mappage des fichiers, autrement dit, spécifier où se trouvent les fichiers décrits dans le manifeste (comme les logos) : via un fichier de mappage ou automatiquement.
Le fichier de mappage s'apparente à ceci :

```
[Files]
"D:\MyManifest.xml"    " AppxManifest.xml"
"\\mon_logo.png"   "logo.png"
```

Il suffira ensuite d'exécuter la commande :

```
> MakeAppx.exe pack /f FichierDeMappage /p MonPackage.appx
```

Pour la méthode automatique, il suffit de créer la hiérarchie de dossiers/fichiers que l'on souhaite, puis d'exécuter la commande suivante :

```
> MakeAppx.exe pack /d DossierMappage /p MonPackage.appx
```

Signature du package

Dernière étape, la signature du package qui est rendue obligatoire par l'utilisation de la commande Add-AppxPackage servant à installer un package. Il faut bien entendu créer un certificat de manière tout à fait classique (.cer et .pvk) que l'on appliquera de cette façon :

```
> MakeCert.exe -r -h 0 -n "CN=Putty Corp" -eku 1.3.6.1.5.5.7.3.3 -pe -sv ClePrivee
.pvk Certificat.cer
> pvk2pfx.exe -pvk ClePrivee.pvk -spc Certificat.cer -pfx ClePfx.pfx
> signtool.exe sign -f ClePfxpfx -fd SHA256 -v .\MonPackage.appx
```

Utilisation des APIs WINRT

Il est déjà possible d'utiliser des APIs WINRT dans une application WPF (je vous invite à visiter notre blog pour un exemple complet de cette méthode) mais une application ainsi convertie pourra alors utiliser un spectre plus large d'API WinRT.
La liste complète est disponible sur le site de Microsoft : https://msdn.microsoft.com/en-us/windows/uwp/porting/desktop-to-uwp-supported-api#new

CONCLUSION

Le Desktop App Converter est, comme vous avez pu le voir, un outil formidable permettant de profiter assez facilement du nouveau packaging des applications sous Windows 10 et donc de son Store.
Aussi, cela permet d'utiliser de nouvelles APIs jusque-là réservées aux applications UWP. Microsoft nous met donc à disposition un formidable outil pour migrer petit à petit nos applications « historiques » vers le SDK UWP.

•

Le **management** en couleurs avec le DISC Partie 2

Thierry Leriche-Dessirier
Consultant freelance
http://www.icauda.com

Votre chef revient d'un séminaire dédié aux outils du manager. Il est enthousiaste. Il ne parle plus que de profils en couleur ; des rouges, des jaunes, des verts ou encore des bleus. Il vous explique que ça va révolutionner la communication entre les membres de l'équipe et la rendre plus efficace. Vous voulez y croire mais cela vous semble bien mystérieux…

Les oppositions

Les profils sont répartis sur la roue DISC de telle sorte que deux profils contigus partagent un certain nombre de caractéristiques tandis que les profils opposés sont l'inverse l'un de l'autre.

Dominant Vs Stable

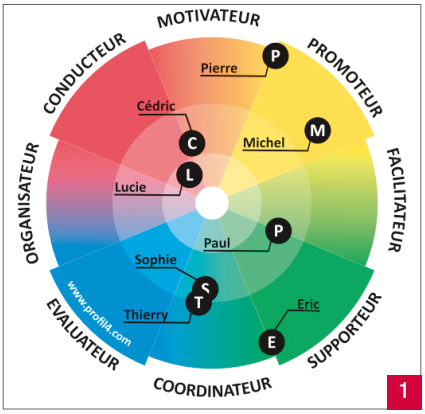
La communication entre le dominant et le stable est complexe et dangereuse. Le dominant parle fort, sans pause. Quand il parle à un stable, il va l'écraser et le stable va se laisser faire. Le dominant va alors penser qu'il a gagné (le dominant ne réalise pas que lorsqu'il gagne, un autre perd) puisque le stable ne dit rien, signe qu'il est d'accord et qu'il a compris que le dominant a raison. Au contraire, le stable se dit que le dominant lui crie dessus et l'agresse. Il adopte une position de défense et se replie sur lui-même. Le dominant pense que tout va bien alors que c'est la catastrophe. A l'opposé, lorsqu'un stable s'adresse à un dominant, il parle lentement. Il fait même des pauses dans ses phrases sans que cela indique qu'il a fini de parler, ce qui est insupportable pour un dominant. On conseille souvent aux dominants de compter jusqu'à cinq avant de reprendre la parole pour être certain de ne pas couper la parole. Le stable parle de sentiments et de feeling avec les gens, ce qui ne veut rien dire pour le dominant. Le stable s'assure que les décisions conviennent aux personnes alors que c'est bien le cadet des soucis du dominant.

Consciencieux Vs Influent

La relation entre les influent et les consciencieux est d'un autre ordre. Ces deux-là se détestent clairement, mais ce n'est pas grave. Le consciencieux considère l'influent comme un beau parleur, un flambeur, un prétentieux, etc. L'influent quant à lui voit le consciencieux comme un pète-sec, qui ne sait pas travailler en équipe, qui ne parle pas, etc. Ils sont sur des longueurs d'ondes différentes mais il ne faut pas s'en inquiéter.

COMMENT S'EN SERVIR ?

L'enjeu du DISC est de mieux communiquer ou, dit autrement, de transmettre de l'information de manière efficace. Cela va se traduire de plusieurs façons selon le contexte. Pour commencer, on pensera au commercial vantant les mérites de ses produits à un client. En connaissant son propre profil et en identifiant celui de son interlocuteur, il sera en mesure d'adapter son discours pour que ses arguments trouvent une oreille attentive, ou plutôt réceptive. Il existe d'ailleurs des variantes du modèle DISC spécialisées pour la vente et le commerce. Attention ici, on pourrait s'imaginer à tort qu'il s'agit d'une forme de manipulation. Ce n'est absolument pas le cas et il est important de le comprendre. Le modèle ne valorise pas les arguments ou leurs justifications. Il en améliore seulement la transmission.



1

Le travail en groupe sera certainement celui qui profitera le plus du modèle. En effet, les membres de l'équipe, ayant des profils différents, vont avoir des façons différentes de comprendre les objectifs et de les atteindre. Or, s'il y a bien quelque chose de commun à l'équipe, ce sont les objectifs. Chaque profil communiquant d'une façon propre, on pourrait s'imaginer des divergences fictives donnant lieux à des conflits bien réels, eux. En partageant au sein du groupe sur les profils de chacun, les membres de l'équipe sont en mesure de se comprendre mutuellement.

Un chef de projet informatique rapportait l'anecdote suivante. Il manageait une équipe d'une demi-douzaine de développeurs. L'ambiance était relativement bonne sauf avec une personne dont le travail était de qualité mais qui se retrouvait en permanence en conflit avec les décisions du groupe. Cela se ressentait dans la communication mais également, et de plus en plus, dans le travail produit. Dans une démarche qualité, l'équipe a passé des tests DISC. Les résultats montraient que la personne avait un profil Influent (jaune) alors que le reste de l'équipe se positionnait à mi-chemin entre les profils Stable (vert) et Consciencieux (bleu), ce qui est courant dans ce métier. Une matinée avait alors suffit pour expliquer au groupe le fonctionnement de chacun et pour mettre en évidence que leurs conflits grandissants n'avaient aucune base sérieuse. Mieux encore, le processus avait pointé du doigt leurs complémentarités et comment en profiter.

Roue des tendances de groupe

Un DRH s'aidera du modèle pour renforcer une orientation de l'équipe [Fig. 1] ou, au contraire, pour la compléter. L'un ou l'autre de ces choix dépendra bien entendu du contexte. Recruter des personnes ayant un profil similaire à celui de l'équipe garantit (en partie) une intégration facile et rapide. Cela renforcera d'autant les qualités appréciées chez l'équipe, mais également ses faiblesses. Au-delà d'une taille critique (5-6 personnes), on estime qu'il est plus avantageux de composer l'équipe avec des profils dispersés sur la roue DISC.

La connaissance des profils de son équipe sera également utile au manager pour affecter les tâches. Cela se traduit non seulement du point de vue de la communication mais également dans le processus et son suivi. Un CdP sera ainsi bien inspiré de traiter chaque membre de l'équipe individuellement, en s'adaptant à ses besoins et particularités. On croise encore trop souvent des CdP qui s'adressent à des équipes et non à des individus, comme si tout le monde fonctionnait de (sa) même façon. En faisant l'effort d'adapter sa communication naturelle à ses interlocuteurs, un CdP obtiendra généralement de meilleurs résultats. Cela peut sembler trivial mais ne l'est pas. Le temps qui y sera consacré sera largement rentabilisé. Par exemple, un Dominant (rouge) préfèrera recevoir des tâches précises, sans entrer dans les détails et, si possible, qui incluent des challenges, même petits.

Exemple de questions

Un Influent (jaune) recherchera la nouveauté et le contact. Par nature, il faudra lui confier une seule mission à la fois. Un Stable (vert) aura besoin de comprendre le contexte dans son ensemble. Il faudra lui donner des explications et lui laisser du temps pour les digérer. Enfin, le Consciencieux (bleu) voudra du factuel et des (nombreux) détails. Si possible, il faut lui indiquer les sujets à l'avance. Il préfèrera travailler en autonomie, avec méthode et sans distraction extérieure.

ETABLIR SON PROFIL ET CELUI DE SON ÉQUIPE

Quand on travaille depuis longtemps avec ses collègues et qu'on a appris à les connaître, il est relativement simple d'en identifier le profil DISC adapté grâce aux nombreux indices qu'on a pu récolter avec le temps. Le profil naturel reste, quant à lui, plus délicat à déterminer puisqu'il est caché par nature. Établir les profils de son équipe par la simple observation, surtout pour une équipe jeune dont les membres ne se connaissent que peu, est complexe. De nombreuses sociétés se tournent donc vers des tests professionnels.

Un manager doit non seulement proposer un test aux membres de son équipe mais aussi le passer lui-même si ce n'est pas déjà fait. On conseille aux managers et aux formateurs de consacrer un peu de temps avant les tests pour les démystifier. Il s'agit d'expliquer les enjeux sans entrer dans les détails et de couper court aux craintes légitimes. Le modèle DISC aide les personnes et les équipes à mieux communiquer et interagir. Ça ne mesure pas le QI, les capacités, et encore moins la santé mentale ou les valeurs. On trouve facilement des sites permettant d'établir des profils DISC sur le Web. On pourra conseiller profil4.com dont sont issues les illustrations de cet article. Ce site propose un test DISC Essentiel gratuit, sans pub et en français, idéal pour une découverte du modèle. Le site offre aussi de la documentation : vidéos, mémentos, médias, etc. Les entreprises, groupes et formations, pourront opter pour une formule avancée. Les lecteurs de Programmez bénéficient de 15% de réduction sur la boutique de profil4.com à l'aide du code « PGMZ15 ».

On partagera les résultats avec les membres de l'équipe. Chacun pourra ainsi se positionner, non seulement sur les roues DISC, mais aussi vis-à-vis des autres membres. À ce stade, on pourra passer une heure ou deux pour approfondir et s'approprier le modèle. Les membres de l'équipe en profiteront pour analyser leurs points communs et leurs divergences, sources de conflits et d'incompréhensions mais aussi de richesse, de diversité et de complémentarité.

Dans la forme, les sites et organismes proposant des tests DISC tentent de se différencier sur l'ergonomie. Sur le fond toutefois, la plupart des tests se ressemblent. Ils demandent généralement de classer des propositions par ordre de préférence [Fig. 2]. Chaque réponse, ou groupe de réponses, est un indice supplémentaire pour établir une orientation.

Pour les créateurs de tests, une des grandes difficultés est de proposer un jeu de questions varié et équilibré. Un autre grand challenge est de maintenir la concentration des utilisateurs sur un nombre assez important de questions pour garantir une précision satisfaisante des résultats. En effet, avec trop peu de questions, les résultats ne seront pas assez fiables pour les exploiter. En augmentant le nombre de questions, on augmente mécaniquement la précision mais on s'expose au risque que l'utilisateur s'ennuie et réponde au hasard pour en finir rapidement. La zone comprise entre 25 et 50 questions est optimale. On réservera toutefois la limite haute à un environnement calme et contrôlé, comme dans le cadre d'un rendez-vous RH.

CONCLUSION

Bien que très incomplète, cette présentation du modèle DISC (Dominant, Influent, Stable et Consciencieux) est suffisante pour en comprendre les principes fondamentaux. Le modèle est né il y a un siècle aux Etats Unis et continue d'évoluer. Il est très utilisé dans les pays anglo-saxons et se démocratise dans nos contrées depuis une quinzaine d'années, en particulier dans les grosses entreprises et les start-ups. Les retours, quant à eux, sont excellents.

Le Web regorge de ressources traitant du modèle DISC ou de sujets connexes. La plupart des sites sont en anglais mais on en trouve quelques-uns en français, comme profil4.com dont on a déjà parlé précédemment et qui offre divers médias à télécharger. La page Wikipédia dédiée au modèle, quant à elle, propose une bonne introduction. On recommandera le site anglais Manager-Tools.com qui fait référence. Sa déclinaison francophone, OutilsDuManager.com, propose un forum, des ressources, des formations et conférences, ainsi qu'un podcast audio bimensuel de qualité, gratuit et en français, traitant du management. Comme souvent, on trouvera de nombreuses vidéos intéressantes sur Youtube. Let's Work (goo.gl/RxafqK), une chaîne francophone dédiée à la vie en entreprise, présente le modèle DISC et une description des quatre composantes à travers une série de vidéos rythmées. Youtube, encore, propose des conférences sur l'agilité et/ou le développement et/ou le management qui abordent le modèle sous différents angles. Avant de se documenter plus, il sera toutefois judicieux de passer un test DISC, sous peine d'en biaiser les résultats plus tard. C'est gratuit, facile et rapide… On pourra s'amuser, à l'aide de la documentation, à trouver des indices chez ses collègues. •

Liens

Profil 4 : https://profil4.com/
Manager-Tools : https://www.manager-tools.com/
Outils du Manager : http://www.outilsdumanager.com/
Let's Work : https://goo.gl/RxafqK
Illustrations par Mimika : http://mimikaweb.fr/

MODÉLISATION DES DONNÉES

Partie 3

Quels intérêts et usages pour le reverse engineering ?

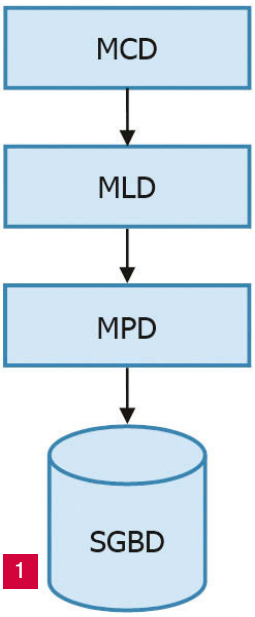
Steve Berberat
Collaborateur scientifique
Haute école de gestion Arc, HES-SO
steve.berberat@he-arc.ch

Pierre-André Sunier
Professeur HES
Haute école de gestion Arc, HES-SO
pierre-andre.sunier@he-arc.ch

En qualité de développeur, il nous est parfois demandé de reprendre une application faite sur mesure que nous ne maîtrisons pas, soit parce que nous n'avons pas participé à son élaboration, soit parce que les derniers travaux sont trop lointains.

Un des premiers challenges est alors de comprendre la base de données sur laquelle l'application s'appuie. Dans ce contexte, le manque de documentation ou sa désuétude s'avèrent être un problème récurrent. Si la base de données a été créée à partir de modèles, alors ces derniers sont très utiles pour appréhender rapidement la structure des données. Que faire si ce n'est pas le cas ? Une solution efficace consiste à utiliser l'ingénierie inverse, appelée aussi rétro-ingénierie et connue sous le nom anglophone « reverse engineering ».

L'ingénierie inverse ne sert pas uniquement à la rétrodocumentation. Une fois réalisée, elle permet de mettre à jour une base de données automatiquement suite à des adaptations faites sur les modèles, ou encore de migrer la structure vers un autre système de gestion de bases de données. Dans cet article, nous allons vous présenter les cas d'usages pour lesquels il y a un intérêt à utiliser l'ingénierie inverse à partir d'une base de données. Nous allons ensuite illustrer son fonctionnement au travers de 2 outils différents.



1 *Modèle conceptuel (MCD), logique (MLD) et physique (MPD) de données*

Rappel sur les modèles de données

La compréhension de l'ingénierie inverse nécessite au préalable de saisir le sens des 3 principaux modèles de données qui peuvent exister, à savoir : le modèle conceptuel de données (MCD), le modèle logique de données (MLD) et le modèle physique de données (MPD). La Figure 1 montre ces 3 modèles, en partant du plus abstrait situé en haut et en allant jusqu'au plus concret situé en bas. Dans le cas où une base de données a été générée automatiquement à partir de modèles, le concepteur est idéalement passé par ces 3 étapes en utilisant un automate qui lui a permis de passer d'un modèle à l'autre de façon automatique, comme le montrent les flèches de la Figure 1.

Modèle conceptuel de données

Le MCD a un haut niveau d'abstraction ; entendez par là qu'il représente la structure des données sans se soucier d'une technologie ou d'un constructeur de base de données particulier. Dans une documentation, ce modèle est très utile, car il permet de se concentrer sur l'essentiel sans se soucier des particularités techniques ! Dans le MCD, nous retrouvons notamment les entités, leurs attributs et les associations entre entités.

Modèle logique de données

Le MLD est propre à une technologie comme, par exemple, la base de données relationnelle, la base de données objet ou même la structure de fichiers XML. Ces différentes technologies étant standardisées, le MLD n'est pas dépendant d'un fournisseur. Dans un MLD propre aux bases de données relationnelles, nous retrouvons les informations propres à la norme ANSI SQL : les tables et leurs colonnes, les différentes contraintes de clé primaire et étrangère, les contraintes d'unicité, les contraintes « NOT NULL », etc.

Modèle physique de données

Le MPD est propre à un constructeur et comprend toutes les informations qui lui sont spécifiques et qui peuvent être déployées vers une base de données réelle. Par exemple, pour le modèle physique d'une base de données Oracle, nous retrouverons les types de colonnes « Varchar2 » et « Number », les tablespaces, les triggers et paquetages, etc. Le concepteur qui a généré la base de données à partir de modèles a utilisé un automate qui lui a automatiquement créé le code SQL à partir du MPD, sans nécessiter une écriture du code à la main.

Le fonctionnement du reverse engineering

L'ingénierie inverse permet de reconstruire les différents modèles à partir d'une base de données existante. Sur le schéma de la Figure 2, ces étapes de reconstruction sont de couleur bleue. Dès lors que les modèles sont obtenus, le concepteur peut, dans une seconde phase, les modifier et utiliser un outil pour, en finalité, mettre à jour la base de données en fonction des modifications spécifiées. Cette phase, que nous pouvons nommer réingénierie, se décompose selon les étapes coloriées en orange sur le schéma.

Les outils qui permettent de réaliser de l'ingénierie inverse, de la génération de bases de données ou de la réingénierie sont connus sous le nom d'atelier de génie logiciel (AGL). Ils sont aujourd'hui souvent intégrés aux environnements de développement spécifiques aux constructeurs.

Les étapes en passant par le MCD

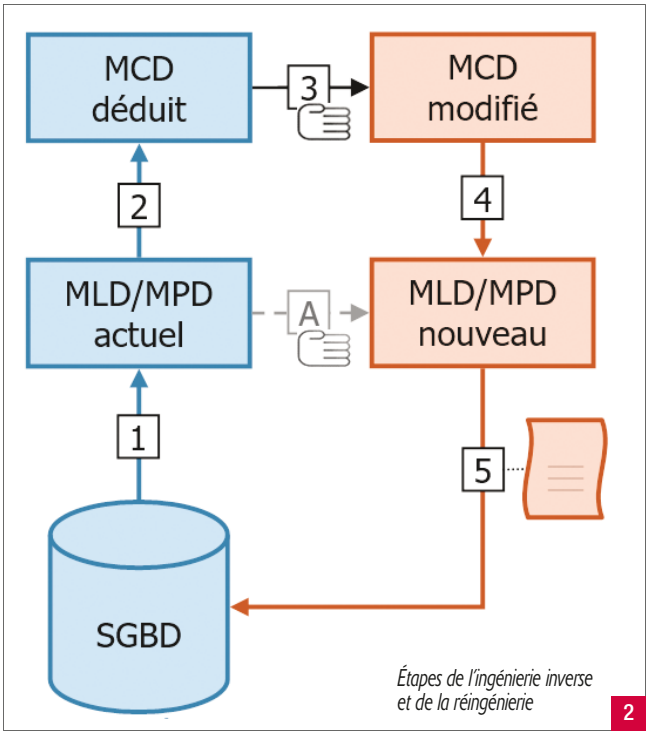
Dans l'étape 1 de la Figure 2, l'AGL se charge de lire le dictionnaire de la base de données et de générer les MLD et MPD. Les deux modèles ont été volontairement regroupés, car les outils d'ingénierie inverse les produisent généralement simultanément.

L'étape 2 consiste à déduire un modèle conceptuel à partir du modèle logique, à l'aide de règles de déduction spécifiques. L'avantage d'obtenir un MCD, au lieu de s'en tenir au MLD, est de disposer d'un modèle qui s'abstrait de la technologie. Il représente ainsi purement le métier et oriente la réflexion uniquement sur la conception.

Si le concepteur souhaite faire évoluer la base de données, il réalise l'étape 3, c'est-à-dire qu'il apporte les modifications souhaitées au modèle conceptuel. Le MCD modifié devient alors la nouvelle référence qui va permettre au concepteur de générer les modèles logiques et physiques automatiquement à l'aide de l'AGL. Ceci correspond à l'étape 4. En dernier lieu, à savoir à l'étape 5, l'AGL produit le code SQL de mise à jour de la structure de données et l'exécute sur la base de données cible. En plus du code SQL, du code spécifique au constructeur peut être accompagné comme, par exemple, des triggers et paquetages PL-SQL dans le cas d'une base de données Oracle.

Les étapes en passant par le MLD

Le passage par le MCD n'est pas obligé. S'il n'y a pas de souhait de disposer d'un modèle conceptuel, alors le concepteur peut s'en tenir au modèle logique. Dans le cas où une adaptation de la base de données est souhaitée, c'est alors l'étape A de la Figure 2 qui est réalisée : le MLD est modifié manuellement selon les besoins. L'AGL peut ensuite reprendre la main pour réaliser la génération du code et le déploiement.



Étapes de l'ingénierie inverse et de la réingénierie 2

Les cas d'usages du reverse engineering

Il existe plusieurs situations différentes qui rendent l'ingénierie inverse utile ou nécessaire. Nous vous proposons ici 5 cas d'usages dans lesquels vous risquez de vous trouver un jour, et où l'ingénierie inverse pourrait réellement répondre à votre besoin.

Cas d'usage 1

Dans cette situation, vous devez faire évoluer une base de données existante, soit pour la corriger, soit pour l'étendre. Vous utilisez alors un AGL pour réaliser les étapes décrites en Figure 2, en passant soit par l'étape 3, soit par l'étape A.

Cas d'usage 2

Vous devez comprendre la structure d'une base de données alors qu'aucune documentation n'existe. Il s'agit ainsi de faire de la rétrodocumentation. L'ingénierie inverse vous permettra alors de créer très rapidement un MLD ou un MCD pour vous offrir une vue d'ensemble et pour vous rendre plus véloce dans l'appréhension de la structure. Ce sont uniquement les étapes 1 et 2 de la Figure 2 qui sont réalisées dans ce cas.

Cas d'usage 3

Vous devez mettre à jour une documentation qui ne l'est plus suite à des adaptations et évolutions faites dans la base de données. Dans ce contexte, les outils d'ingénierie inverse prennent généralement en charge la mise à jour de modèles existants lors de la réalisation des étapes 1 et 2 de la Figure 2.

Cas d'usage 4

Vous devez migrer votre base de données vers une autre instance de base de données ou vers le système d'un autre constructeur. L'ingénierie inverse vous permettra alors de récupérer la structure et de la déployer vers la nouvelle base de données cible. Comme pour le cas d'usage 1, toutes les étapes sont réalisées. La différence est que, à l'étape 5, les paramètres d'accès vers la nouvelle base de données seront renseignés à l'AGL.

Cas d'usage 5

Vous devez réaliser un audit qualité sur une base de données existante. La rétro-ingénierie vous permettra alors de créer des modèles et de vérifier si le niveau de qualité des données souhaité est assuré, ou si la documentation existante correspond bien à la réalité. •

Suite et fin le mois prochain

Références

Projet MVC-CD et téléchargement de l'automate de transformation : http://lgl.isnetne.ch/Sagex35793/index.htm

Droz-dit-Busset Michaël que nous remercions, Travail de Bachelor « MVC-CD – Ingénierie inverse », juillet 2015

Présentation des **JSON Web Tokens** (JWT) ou comment sécuriser une API Web ?

• Antoine PACAUD
Directeur technique – Technologies PHP & Front-end chez
Webnet

Les web-services sont aujourd'hui au centre des SI de nos entreprises. Ils permettent de transmettre l'information entre des systèmes hétérogènes, à la fois au sein de l'entreprise mais également à l'extérieur, notamment, et, de plus en plus, vers des systèmes mobiles. Or exposer cette information, potentiellement critique, pose la question de la sécurisation de ces échanges.

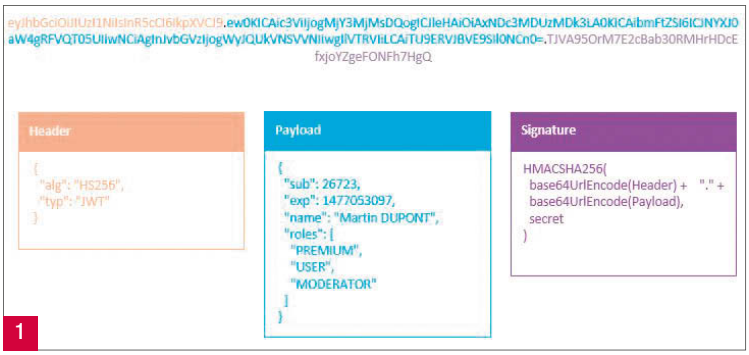
Une première étape essentielle est de recourir à l'utilisation de HTTPS pour l'ensemble des échanges vers l'API. Cela permettra de parer aux attaques les plus répandues en empêchant la lecture des données transmises et reçues (comme le couple login/mot de passe d'un utilisateur par exemple).
Reste ensuite à trouver un mécanisme d'authentification des échanges entre le client et l'ensemble des web-services qui soit robuste et qui s'intègre facilement avec l'ensemble des types de clients possibles.
L'une des spécificités de l'authentification sur une API est que celle-ci est « stateless » (sans état). Cela implique que, contrairement au mode « Web » classique qui requiert souvent l'utilisation d'une session, le serveur ici ne conserve pas de « contexte » ou d'informations sur l'utilisateur du service en mémoire. L'utilisateur est uniquement authentifié pour la durée de la transaction, et aucune information temporaire ne subsiste à la fin de celle-ci. Ainsi, toutes les données nécessaires à l'authentification doivent être passées à chaque fois. Cela permet également de mettre en place de multiples serveurs frontaux sans avoir à partager de sessions entre eux, et rend ainsi l'application plus scalable.

Présentation de JWT

JWT ou JSON Web Token est un standard ouvert désormais industriel (RFC 7519) qui tend à répondre à ces problématiques. De plus en plus utilisé il dispose d'une communauté active et de librairies pour plus d'une vingtaine de langages. Pour comprendre son fonctionnement, prenons l'exemple d'une application mobile sur laquelle l'utilisateur doit s'authentifier afin d'accéder à des ressources exposées par un ensemble de web-services (API). La première étape consiste donc pour l'utilisateur à envoyer son couple login/mot de passe via un service d'authentification.
Il n'est pas envisageable de stocker ce couple en clair sur le device de l'utilisateur et d'envoyer celui-ci à chaque échange suivant pour des raisons de sécurité. Nous avons donc besoin de stocker un artéfact qui servira à authentifier l'utilisateur auprès de l'API.
C'est ici que rentre en jeu JWT :

- Le serveur reçoit le couple login/mot de passe lors du premier échange ;
- Si ce couple est valide, le serveur génère un token permettant d'identifier l'utilisateur pour les prochains échanges vers l'API et le renvoie au client ;
- Le client stocke le token en local (via le LocalStorage par exemple) ;
- Ce token est retransmis au serveur pour chaque appel suivant, permettant ainsi au serveur d'identifier l'utilisateur.

Regardons un peu plus en détails le token généré. Celui-ci est une suite de caractères alphanumériques décomposables en 3 parties séparées par un point. [Fig.1]



Décomposition d'un token JWT

1ère partie : Header

La première partie est un objet JSON encodé en Base64 qui représente l'entête du token.
Il décrit l'algorithme utilisé pour la signature du token (voir plus loin) ainsi que le type du token (JWT ici).

2e partie : Payload

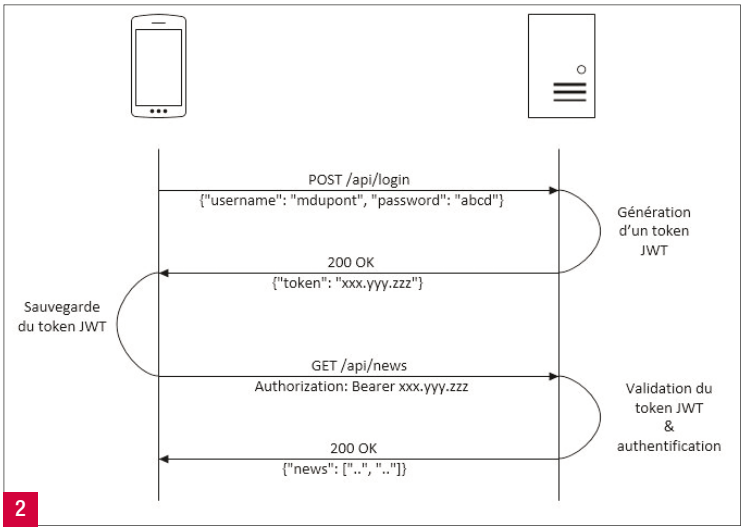
La seconde partie est également un objet JSON encodé en Base64 qui représente cette fois-ci le corps (ou payload) du token. Celui-ci contient des informations supplémentaires sur le token (comme sa date d'expiration) mais également et surtout à destination de l'application. C'est dans cet objet qu'on placera, par exemple, l'identifiant de l'utilisateur courant, son niveau de droit, ou toute information que l'on souhaite récupérer rapidement sans avoir à effectuer de requêtes supplémentaires.
Le standard définit différents types de propriétés (appelées « claims ») réservées, publiques et privées. Les propriétés réservées et publiques sont normalisées sur 3 caractères afin de réduire la taille finale du token. On retrouvera notamment :

- **sub :** définit le sujet du token, en règle générale un identifiant d'utilisateur ;
- **exp :** définit la date et l'heure (timestamp) à laquelle le token expire (voir expiration plus loin).

Vous pouvez également ajouter autant de propriétés privées que vous le souhaitez en fonction de vos besoins.

3e partie : Signature

La dernière partie du token est sa signature.
En effet, et c'est l'une des forces de JWT, cette dernière partie permet de s'assurer que le contenu des 2 premières n'a pas été dénaturé par un utilisateur malveillant. Le token étant envoyé par le client, on ne peut pas lui



Chronologie des échanges client ó serveur

faire confiance aveuglément. La signature est un hash réalisé sur la concaténation des 2 premières parties via un algorithme précisé dans le header du token (HMACSHA256 via un secret ou RSASHA256 via une clé publique et une clé privée). Le serveur va donc valider le token à chaque transaction afin de s'assurer que sa signature est valide. Ainsi, si le client modifie le contenu du corps du token (pour obtenir un niveau de droit plus élevé par exemple), la signature n'étant plus correcte, le token sera rejeté.

Transmission du token

Reprenons donc le déroulé des transactions. [Fig.2]
Le token est donc transmis dans les headers de chaque requête via la propriété « Authorization » de la manière suivante :
Authorization : Bearer xxxxx.yyyyyy.zzzzzz

Gestion de l'expiration d'un token

La seconde question que l'on doit se poser désormais est la durée pendant laquelle ce token doit être valide. Pour répondre à cette question, il faut imaginer que ce token se retrouve entre des mains malintentionnées et se demander combien de temps il est acceptable de lui laisser l'accès à vos informations via ce token. Ce délai dépend donc réellement du caractère plus ou moins sensible des données échangées, de l'environnement dans lequel les web-services sont consommés mais également du moyen qui sera utilisé pour le renouveler.
En effet, si la consommation se réalise dans un environnement interne uniquement et isolé de l'extérieur, on pourra imaginer une durée de validité bien plus longue que si les échanges se font via Internet et donc sont bien plus exposés aux attaques et interceptions.
De même, un renouvellement de token engendre un trafic réseau plus intense car il nécessite de nouveaux échanges. Dans un environnement mobile, on tentera de trouver un juste milieu pour limiter au maximum ces échanges supplémentaires.
En moyenne, on trouvera une durée entre 5 minutes et une journée, et une heure peut sembler un bon compromis dans le cadre du mobile.

Une fois la durée déterminée, on utilisera la propriété réservée « exp » dans le corps du token pour y insérer le timestamp de la date d'expiration de celui-ci. La plupart des librairies qui implémentent JWT contrôleront automatiquement cette date à chaque requête et généreront une erreur 401 si le token a expiré pour indiquer au client qu'il doit à nouveau récupérer un token valide.
Il faut donc trouver un moyen désormais pour que le client obtienne un nouveau token.

Les « refresh tokens »

Comme indiqué plus haut, notre application ne doit pas conserver le login/mot de passe de l'utilisateur, elle ne conserve que le token issu de l'authentification initiale de l'utilisateur. Une fois que celui-ci expire, il faudrait donc demander à l'utilisateur de se ré-authentifier. Si cela est acceptable sur des applications critiques (applications bancaires par exemple), cela s'avère toutefois très contraignant pour l'utilisateur, à fortiori si la durée d'expiration du token est rapide. Une solution alternative consiste à utiliser un second jeton qu'on appelle « refresh token ».
Lors de l'authentification initiale de l'utilisateur, le serveur génère un token aléatoire. Celui-ci va être attaché au compte client (en base de données par exemple), et renvoyé au client en complément du token JWT. [Fig.3].
Ce token devra être d'une longueur suffisante pour éviter toute attaque de type brute-force.
Lorsque le jeton JWT expire, notre application pourra envoyer une requête sur une url particulière en transmettant ce token pour obtenir en retour un nouveau jeton JWT valide ainsi qu'un nouveau « refresh token ». En effet, le refresh-token est à usage unique afin de réduire encore plus les risques d'attaque de type brute force.
On pourra également ajouter une date d'expiration à ce refresh-token (par exemple une semaine).
Si le jeton JWT expire et que le refresh-token n'est plus valide, l'application exigera alors une nouvelle authentification classique de l'utilisateur via son login et mot de passe pour recommencer un cycle.

EN CONCLUSION

JWT est un standard qui tend à s'imposer pour l'authentification. Simple de mise en œuvre lors du développement du client, il l'est tout autant côté serveur grâce aux nombreuses librairies à disposition. Son format compact et le choix du format JSON lui permettent d'être utilisé dans tous les contextes contrairement à SAML qui utilise du XML et, de ce fait, est beaucoup plus lourd en termes de développement et de poids.
S'il sert ici à l'authentification, il peut également être utilisé pour échanger des informations de manière sécurisée entre 2 parties qui pourront s'assurer de la validité et de la provenance de ces dernières en échangeant la clé de signature.
Séduits ? Il ne vous reste plus qu'à tenter de l'implémenter pour votre prochaine API !
Informations utiles sur le site officiel : https://jwt.io •

```
{
   "token":"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.ew0KICAic3ViIjogMjY3MjMsDQogICJl...",
   "refresh_token":"ICJNYXJ0aW4gRFVQT05UIgInJvbGVzIjoI1NiIsCJNYXJ0aW4gRFVQT05UIiwInRkVNSVVNIiwgIlVTR5..."
}
```
3

Retour du service d'authentification

Forensic Python

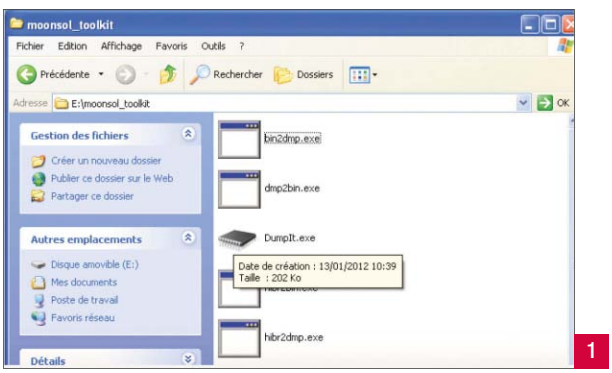
Franck Ebel, *Expert R&D et Formations Serval-concept / serval-formation Responsable Licence CDAISI, UVHC Commandant de Gendarmerie réserviste, cellule Cyberdéfense*

La mémoire vive contient toutes les données qui sont traitées lors du démarrage de l'ordinateur. Elle agit comme une sorte de disque dur à accès rapide ; plus sa capacité est importante plus les performances de la machine sont élevées en termes de rapidité d'exécution des programmes. Contrairement au disque dur, la mémoire RAM est sollicitée pour chacune des actions réalisées sur le système. Elle contient donc plus de données sensibles telles que des mots de passe saisis, un historique, des données non maîtrisées et non inscrites sur le disque dur...

Il existe de nombreux moyens pour dumper la mémoire, ce qui en fait une cible privilégiée. La compromission d'un système par l'altération directe de la mémoire RAM est plus discrète que celle réalisée par l'altération des fichiers du disque dur. Contrairement à certains disques durs, le contenu de la mémoire RAM n'est pas chiffré et facilite ainsi l'accès au système ou sa manipulation même si le disque dur est chiffré. La solidité d'une chaîne est conditionnée par celle de son maillon le plus faible. C'est de cette manière que le vol d'un seul ordinateur portable peut entraîner des pertes colossales : des données sensibles sont stockées sur ces PC.
Les experts en sécurité informatique recommandent le chiffrement des données, donc du disque dur. La sécurité des données dans ce cas repose sur le fait que la mémoire vive est effacée dès qu'elle n'est plus sous tension. La seule façon de récupérer alors la clé de cryptage est de contrôler la machine. Il existe de nombreux logiciels permettant l'extraction de la mémoire RAM lorsqu'une session Windows est démarrée et contrôlée. Ils nécessitent tous les droits Administrateur pour s'exécuter. Leur exécution peut être réalisée à distance via un accès obtenu licite (accès Netbios ou VNC par exemple) ou illicite (via une faille identifiée et exploitée), et ne nécessite pas d'avoir obligatoirement un accès physique à la machine. Ils peuvent tous être placés sur un média amovible afin de réaliser une extraction depuis celui-ci sans qu'aucun outil ou librairie n'ait besoin d'être installé sur le système. Dans la suite, nous devons avoir des droits d'administration, nous partirons du principe que le système Windows est compromis.

Dump mémoire

La première chose à effectuer est un dump mémoire, c'est à dire récupérer le contenu entier de la mémoire avant de l'exploiter grâce au logiciel en Python Volatility que nous verrons dans le point suivant. Comment effectuer un dump mémoire, nous allons voir un moyen ; il en existe beaucoup, de nombreux logiciels existent. Depuis 2011, la suite « MoonSol 's Windows Memory Toolskit » de Matthieu Suiche propose un exécutable permettant de dumper la mémoire à partir d'une clé usb : Dumpit.exe. Le fonctionnement en est très simple : cliquer sur l'exécutable et suivre les indications. [Fig.1]. Il suffit donc ensuite de lui donner la destination du Dump, sur la clé usb ici, et un nom : IUT_270_MODELE-20120115-095924.raw [Fig.2]. Nous avons donc maintenant un Dump mémoire que nous pouvons utiliser pour la suite.



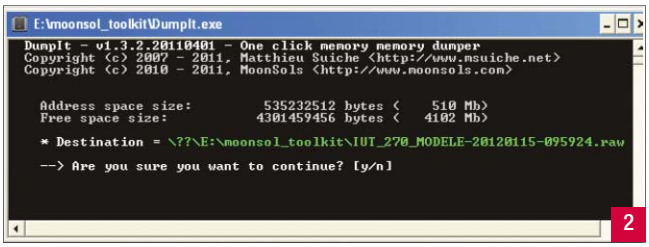
1

Analyse du Dump mémoire.

Volatility est un framework développé en Python qui permet d'analyser une empreinte de mémoire physique (dump mémoire). Il existe deux méthodologies pour réaliser les analyses forensics ; la méthode à chaud (live forensics) et la méthode à froid (dead forensics). Volatility est utilisé dans le cadre de la méthode à chaud. Il suffit ensuite de le décompresser et d'avoir au préalable installé Python. Volatility fonctionne avec des plugins. Pour les voir, la commande suivante vous renseignera sur eux et vous fournira d'autres renseignements :

```
Python vol.py -h
```
[Fig.3]

Il est téléchargeable directement sur le site suivant :
https://www.volatilesystems.com/default/volatility
La dernière version est la 2.31 ; elle est capable d'analyser des dumps mémoire sur xp sp1-2, vista sp0-1-2, seven sp0-1, 2003 sp0-1-2,2008 sp1-2 [Fig.4]



2

```
/volatility-2.0 $ ./vol.py --info
Volatile Systems Volatility Framework 2.0


PLUGIN_COMMANDS
---------------
bioskbd      - Reads the keyboard buffer from Real Mode memory
connections  - Print list of open connections [Windows XP Only]
connscan     - Scan Physical memory for _TCPT_OBJECT objects (tcp connections)
crashinfo    - Dump crash-dump information
dlldump      - Dump DLLs from a process address space
dlllist      - Print list of loaded dlls for each process
driverscan   - Scan for driver objects _DRIVER_OBJECT
filescan     - Scan Physical memory for _FILE_OBJECT pool allocations
getsids      - Print the SIDs owning each process
handles      - Print list of open handles for each process
hashdump     - Dumps passwords hashes (LM/NTLM) from memory
hibinfo      - Dump hibernation file information
hivedump     - Prints out a hive
hivelist     - Print list of registry hives.
hivescan     - Scan Physical memory for _CMHIVE objects (registry hives)
imagecopy    - Copies a physical address space out as a raw DD image
imageinfo    - Identify information for the image
inspectcache - Inspect the contents of a cache
kdbgscan     - Search for and dump potential KDBG values
kpcrscan     - Search for and dump potential KPCR values
lsadump      - Dump (decrypted) LSA secrets from the registry
memdump      - Dump the addressable memory for a process
```
3

Download

Volatility-2.0: tar.gz / zip / standalone EXE / EXE (python installed) / md5 / sha1
Volatility-1.3_Beta: tar.gz zip md5 sha1 gpg tar.gz gpg zip gpg_key
Volatility-1.1.2: tar.gz zip md5 sha1 gpg tar.gz gpg zip gpg_key
Volatility-1.1.1: tar.gz md5 sha1 gpg gpg_key
4

La commande suivante vous affichera l'aide sur la commande « vol.py » ainsi que les différents plugins : [Fig.5]

Si vous désirez une aide sur le fonctionnement spécifique d'un plugin, il suffit d'appeler l'aide sur celui-ci de la façon suivante :

```
Python vol.py <plugin> -h
```
[Fig.6]

Vous commencerez par demander les informations inhérentes au dump mémoire concerné.

```
Python -f <chemin du dump> imageinfo
```
[Fig.7]

Vous savez maintenant que votre dump concerne un Windows xp sp3, il faudra maintenant préciser ce profil pour chaque plugin utilisé. Pour lister les processus présents en mémoire au moment du dump :

5
```
./vol.py -h
Volatile Systems Volatility Framework 2.0
Usage: Volatility - A memory forensics analysis platform.

Options:
  -h, --help            list all available options and their default values.
                        Default values may be set in the configuration file
                        (/etc/volatilityrc)
  --conf-file=/home/rezor/.volatilityrc
                        User based configuration file
  -d, --debug           Debug volatility
  --info                Print information about all registered objects
  --plugins=PLUGINS     Additional plugin directories to use (colon separated)
  --cache-directory=/home/rezor/.cache/volatility
                        Directory where cache files are stored
  --no-cache            Disable caching
  --tz=TZ               Sets the timezone for displaying timestamps
  -f FILENAME, --filename=FILENAME
                        Filename to use when opening an image
  -k KPCR, --kpcr=KPCR  Specify a specific KPCR address
  -g KDBG, --kdbg=KDBG  Specify a specific KDBG virtual address
  --output=text         Output in this format (format support is module
                        specific)
  --output-file=OUTPUT_FILE
                        write output in this file
  -v, --verbose         Verbose information
  --dtb=DTB             DTB Address
  --cache-dtb           Cache virtual to physical mappings
  --use-old-as          Use the legacy address spaces
  -w, --write           Enable write support
  --profile=WinXPSP2x86
                        Name of the profile to load
  -l LOCATION, --location=LOCATION
                        A URN location from which to load an address space
```

```
python vol.py hashdump -h
Volatile Systems Volatility Framework 2.0
Usage: Volatility - A memory forensics analysis platform.

Options:
  -h, --help            list all available options and their default values.
                        Default values may be set in the configuration file
                        (/etc/volatilityrc)
  --conf-file=/home/rezor/.volatilityrc
                        User based configuration file
  -d, --debug           Debug volatility
  --info                Print information about all registered objects
  --plugins=PLUGINS     Additional plugin directories to use (colon separated)
  --cache-directory=/home/rezor/.cache/volatility
                        Directory where cache files are stored
  --no-cache            Disable caching
  --tz=TZ               Sets the timezone for displaying timestamps
  -f FILENAME, --filename=FILENAME
                        Filename to use when opening an image
  -k KPCR, --kpcr=KPCR  Specify a specific KPCR address
  -g KDBG, --kdbg=KDBG  Specify a specific KDBG virtual address
  --output=text         Output in this format (format support is module
                        specific)
  --output-file=OUTPUT_FILE
                        write output in this file
  -v, --verbose         Verbose information
  --dtb=DTB             DTB Address
  --cache-dtb           Cache virtual to physical mappings
  --use-old-as          Use the legacy address spaces
  -w, --write           Enable write support
  --profile=WinXPSP2x86
                        Name of the profile to load
  -l LOCATION, --location=LOCATION
                        A URN location from which to load an address space
  -y SYS_OFFSET, --sys-offset=SYS_OFFSET
                        SYSTEM hive offset (virtual)
  -s SAM_OFFSET, --sam-offset=SAM_OFFSET
                        SAM hive offset (virtual)
```
6

```
python vol.py -f ../../dumps-memory/dump_xp_dom1.dmp imageinfo
Volatile Systems Volatility Framework 2.0
Determining profile based on KDBG search...

          Suggested Profile(s) : WinXPSP3x86, WinXPSP2x86 (Instantiated with WinXPSP2x86)
                     AS Layer1 : JKIA32PagedMemory (Kernel AS)
                     AS Layer2 : WindowsCrashDumpSpace32 (/datas/courssecu/failles-physiques/dumpmemoire/dumps-memory/dump_xp_dom1.dmp)
                     AS Layer3 : FileAddressSpace (/datas/courssecu/failles-physiques/dumpmemoire/dumps-memory/dump_xp_dom1.dmp)
                      PAE type : No PAE
                           DTB : 0x39000
                          KDBG : 0x8054cde0
                          KPCR : 0xffdff000L
            KUSER_SHARED_DATA : 0xffdf0000L
          Image date and time : 2011-03-30 14:13:49
    Image local date and time : 2011-03-30 14:13:49
         Number of Processors : 1
                    Image Type : Service Pack 3
```
7

```
Python –profile=WinXPSP3x86 -f <chemin du dump> pslist
```
[Fig.8]

Le plugin « psscan » fait la même chose avec un peu plus de détails :

```
Python –profile=WinXPSP3x86 -f <chemin du dump> psscan
```
[Fig.9]

Les hashes du système Windows en cours peuvent être retrouvés dans le dump mémoire. Pour les afficher, il faut connaître les adresses physiques des emplacements en mémoire de la syskey et de la base SAM.

On affiche dans un premier temps les adresses des registres présents en mémoire :

```
Python –profile=WinXPSP3x86 -f <chemin du dump> hivelist[
```
[Fig.10]

Il suffit ensuite de demander l'affichage des hashes à l'aide du plugin hash-dump en lui précisant l'adresse en mémoire de la syskey et de la base SAM :

```
Python –profile=WinXPSP3x86 -f <chemin du dump> hashdump -y 0xe1035b60
-s 0xe1490b60
```
[Fig.11]

8
```
python vol.py -f ../../dumps-memory/dump_xp_dom1.dmp pslist
Volatile Systems Volatility Framework 2.0
 Offset(V)  Name                PID    PPID   Thds     Hnds   Time
---------- -------------------- ------ ------ ------ -------- -------------------
0x81fcaa00 System                  4      0     70     2277 1970-01-01 00:00:00
0x81bd13a8 smss.exe              596      4      3       19 2011-03-30 14:02:44
0x81be96b0 csrss.exe             664    596     11      484 2011-03-30 14:02:46
0x81b521c8 winlogon.exe          688    596     19      485 2011-03-30 14:02:46
0x81b68da0 services.exe          732    688     15      281 2011-03-30 14:02:47
0x81b6fda0 lsass.exe             744    688     19      353 2011-03-30 14:02:47
0x81b68568 svchost.exe           920    732     17      201 2011-03-30 14:02:48
0x81b87660 svchost.exe           984    732     11      297 2011-03-30 14:02:48
0x81c34020 svchost.exe          1072    732     63     1216 2011-03-30 14:02:48
0x81718458 SCTSvc.exe           1096    732      9      167 2011-03-30 14:02:48
0x81af8c20 svchost.exe          1304    732      6       89 2011-03-30 14:02:50
0x81b08978 svchost.exe          1356    732     10      155 2011-03-30 14:02:50
0x81b09458 AvastSvc.exe         1416    732     57      744 2011-03-30 14:02:50
0x813d5da0 spoolsv.exe          1744    732      9      108 2011-03-30 14:03:01
0x813dc610 svchost.exe          1864    732      4      107 2011-03-30 14:03:17
0x8163da20 ApplicationUpda      1904    732      3      117 2011-03-30 14:03:17
0x81acdbe0 jqs.exe              2044    732      5      187 2011-03-30 14:03:18
0x812ecda0 ngctw32.exe           232    732      5      116 2011-03-30 14:03:18
0x81638da0 OcsService.exe        268    732      5       69 2011-03-30 14:03:19
0x81358b28 alg.exe              1288    732      6      106 2011-03-30 14:03:24
0x812d75b8 GoogleUpdate.ex      1832    936      4      146 2011-03-30 14:03:30
0x81bfd5e0 explorer.exe          900    472     16      475 2011-03-30 14:06:30
0x81267b38 hkcmd.exe            2288    900      3       93 2011-03-30 14:06:37
0x81385ac0 igfxpers.exe         2372    900      4      101 2011-03-30 14:06:37
0x813cc360 smax4pnp.exe         2284    900      3       90 2011-03-30 14:06:37
0x81e01da0 AdobeARM.exe         2204    900      8      218 2011-03-30 14:06:37
0x812aa5b0 AvastUI.exe          2324    900      8      114 2011-03-30 14:06:37
0x81a8a960 winampa.exe          2252    900      1       31 2011-03-30 14:06:37
0x81e2fda0 jusched.exe          1964    900      5      135 2011-03-30 14:06:38
0x81e0e020 ngtray.exe           2040    900      2       58 2011-03-30 14:06:38
0x81365c68 Bubble.exe           2444    900      4      111 2011-03-30 14:06:38
0x8132b928 ctfmon.exe           2440    900      1       72 2011-03-30 14:06:38
0x8122fbd0 unsecapp.exe         2876    920      3      103 2011-03-30 14:06:44
0x81329638 wuauclt.exe          3368   1072      7      134 2011-03-30 14:11:40
0x81228020 cmd.exe              2708    900      1       34 2011-03-30 14:12:31
0x8138ac18 win32dd.exe          3932   2708      1       23 2011-03-30 14:13:47
```

```
python vol.py -f ../../dumps-memory/dump_xp_dom1.dmp psscan
Volatile Systems Volatility Framework 2.0
 Offset      Name             PID    PPID   PDB        Time created             Time exited
---------- ---------------- ------ ------ ---------- ------------------------ ------------
----
0x01628020 cmd.exe            2708    900 0x05563000 2011-03-30 14:12:31

0x0162fbd0 unsecapp.exe       2876    920 0x1a3bb000 2011-03-30 14:06:44

0x01667b38 hkcmd.exe          2288    900 0x1a75c000 2011-03-30 14:06:37

0x016aa5b0 AvastUI.exe        2324    900 0x1bfbc000 2011-03-30 14:06:37

0x016d75b8 GoogleUpdate.ex    1832    936 0x1a7af000 2011-03-30 14:03:30

0x016ecda0 ngctw32.exe         232    732 0x16b10000 2011-03-30 14:03:18

0x01729638 wuauclt.exe        3368   1072 0x1c714000 2011-03-30 14:11:40

0x0172b928 ctfmon.exe         2440    900 0x14504000 2011-03-30 14:06:38

0x01758b28 alg.exe            1288    732 0x17f9a000 2011-03-30 14:03:24

0x01765c68 Bubble.exe         2444    900 0x0412a000 2011-03-30 14:06:38

0x01785ac0 igfxpers.exe       2372    900 0x17fa1000 2011-03-30 14:06:37

0x0178ac18 win32dd.exe        3932   2708 0x08c10000 2011-03-30 14:13:47
```
9

```
python vol.py -f ../../dumps-memory/dump_xp_dom1.dmp hivelist
Volatile Systems Volatility Framework 2.0
Virtual     Physical    Name
0xe1d7db60  0x13988b60  \Device\HarddiskVolume1\Documents and Settings\LocalService\NTUSER.DAT
0xe1c10b60  0x131ceb60  \Device\HarddiskVolume1\Documents and Settings\NetworkService\Local Settings\Application Data\Microsoft\Windows\UsrClass.dat
0xe1bec6a0  0x12d886a0  \Device\HarddiskVolume1\Documents and Settings\NetworkService\NTUSER.DAT
0xe1436b60  0x0f7eab60  \Device\HarddiskVolume1\WINDOWS\system32\config\software
0xe14366b8  0x0f7ea6b8  \Device\HarddiskVolume1\WINDOWS\system32\config\default
0xe1490b60  0x0f81eb60  \Device\HarddiskVolume1\WINDOWS\system32\config\SAM
0xe17c9758  0x0f2bb758  \Device\HarddiskVolume1\WINDOWS\system32\config\SECURITY
0xe133c288  0x02b90288  [no name]
0xe1035b60  0x02887b60  \Device\HarddiskVolume1\WINDOWS\system32\config\system
0xe1022758  0x0284e758  [no name]
0x8068fd58  0x0068fd58  [no name]
0xe1c93008  0x196a6008  \Device\HarddiskVolume1\Documents and Settings\Administrateur\Local Settings\Application Data\Microsoft\Windows\UsrClass.dat
0xe12e9008  0x0028e008  \Device\HarddiskVolume1\Documents and Settings\Administrateur\NTUSER.DAT
0xe1d62b60  0x19db8b60  \Device\HarddiskVolume1\Documents and Settings\LocalService\Local Settings\Application Data\Microsoft\Windows\UsrClass.dat
```
10

```
python vol.py -f ../../dumps-memory/dump_xp_dom1.dmp hashdump -y 0xe1035b60 -s 0xe1490b60
Volatile Systems Volatility Framework 2.0
Administrateur:500:8fefcb557a836e500f91f0644acd94b4:7bed448fe2fbf1a9f1d6cb4bf21b0443:::
Invit?:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
HelpAssistant:1000:f263a33284ae64116598c48b0d8e68c1:919796b7725438862b6c40a7413941d9:::
SUPPORT_388945a0:1002:aad3b435b51404eeaad3b435b51404ee:5b3852b687894b0bd20ba5a56b765722:::
igi:1003:aad3b435b51404eeaad3b435b51404ee:9cab1178283c1293b149c5e0cf2066c9:::
```
11

Si on se place maintenant dans le cadre d'une intrusion sur un système, l'analyse du dump peut permettre de donner des indices sur celle-ci.
Le plugin « printkey » va permettre de s'intéresser aux valeurs des clés et sous-clés du registre et également aux dernières dates de modification de celles-ci. Il suffit de lancer la commande suivante :

```
Python -f <chemin dump> printkey
```
[Fig.12]

On peut également s'intéresser à une clé plus spécifiquement.
Après avoir affiché les offsets des clés de registre à l'aide du plugin « hivelist » vu précédemment on lance la commande suivante :

```
Python -f <chemin dump> -o <offset clé recherchée> printkey
```
[Fig.13]

A ce niveau, on s'aperçoit que devant chaque sous-clé est écrit un « S » ou un « V » pour « Stable » et « Volatile ». Les données volatiles (en mémoire) nous intéressant nous allons explorer celles-ci à l'aide du commutateur « K ».

```
Python -f <chemin dump> -o <offset clé recherchée> printkey -K <nom sous clé de registre>
```
[Fig.14]

Le dump de la clé LSA (service d'autorité Locale de sécurité pour Local Security Authority) peut également être intéressant (mot de passe).
On utilise alors le plugin « lsadump » en précisant l'offset de la clé « system » et celui de la clé « Security ».

```
Python -f <chemin dump> -o <offset clé recherchée> lsadump -y <offset clé system> -s <offset clé Security>
```
[Fig.15]

Le plugin « hivedump » permet également de dumper un registre. Le dump sera redirigé dans un fichier « . txt » pour relecture facile car assez long.

```
Python -f <chemin dump> -o <offset clé registre> hivedump
```
[Fig.16]

On peut également vouloir s'intéresser aux connexions ouvertes. Le plugin « connections » permet de lister les connexions en cours. Celui-ci est spécifique à XP, pour Seven, Vista, 2003 et 2008, il faudra utiliser le plugin « netscan » . La commande est la suivante :

```
python vol.py -f <chemin du dump> connections
```
[Fig.17]

Le plugin « connscan » nous donnera plus d'informations sur les connexions. On vérifiera que le pid associé avec la connexion est présent dans le résultat de « pslist », dans le cas contraire cela pourrait signifier la présence d'un rootkit. La commande est la suivante :

```
python vol.py -f <chemin du dump> connscan
```
[Fig.18]

Le plugin « socket » permettra de s'intéresser aux sockets ouverts afin de vérifier qu'ils ne sont pas suspects.

```
python vol.py -f <chemin du dump> sockets
```
[Fig.19]

Il peut également être intéressant de connaître les SID des processus en cours pour avoir une idée des utilisateurs. Un SID (identificateur de sécu-

```
python vol.py -f ../../dumps-memory/dump xp dom1.dmp printkey
Volatile Systems Volatility Framework 2.0
Legend: (S) = Stable   (V) = Volatile

----------------------------
Registry: \Device\HarddiskVolume1\Documents and Settings\LocalService\Local Settings\Application Data\Microsoft\Windows\UsrClass.dat
Key name: S-1-5-19_Classes (S)
Last updated: 2010-05-03 09:57:15

Subkeys:

Values:
----------------------------
Registry: \Device\HarddiskVolume1\Documents and Settings\NetworkService\NTUSER.DAT
Key name: $$$PROTO.HIV (S)
Last updated: 2010-05-03 09:54:57

Subkeys:
  (S) AppEvents
  (S) Console
  (S) Control Panel
  (S) Environment
  (S) Identities
  (S) Keyboard Layout
  (S) Printers
  (S) Software
  (S) UNICODE Program Groups
```
12

```
python vol.py -f ../../dumps-memory/dump xp dom1.dmp printkey -o 0xe1035b60
Volatile Systems Volatility Framework 2.0
Legend: (S) = Stable   (V) = Volatile

----------------------------
Registry: User Specified
Key name: $$$PROTO.HIV (S)
Last updated: 2011-03-30 14:02:32

Subkeys:
  (S) ControlSet001
  (S) ControlSet002
  (S) LastKnownGoodRecovery
  (S) MountedDevices
  (S) Select
  (S) Setup
  (S) WPA
  (V) CurrentControlSet

Values:
```
13

```
python vol.py -f ../../dumps-memory/dump xp dom1.dmp printkey -o 0xe1035b60 -K CurrentControlSet
Volatile Systems Volatility Framework 2.0
Legend: (S) = Stable   (V) = Volatile

----------------------------
Registry: User Specified
Key name: CurrentControlSet (V)
Last updated: 2011-03-30 14:02:32

Subkeys:

Values:
REG_LINK      SymbolicLinkValue : (V) \Registry\Machine\System\ControlSet001
```
14

```
python vol.py -f ../../dumps-memory/dump xp dom1.dmp lsadump -y 0xe1035b60 -s 0xe17c9758
Volatile Systems Volatility Framework 2.0
9015040C600011D38

0000   00 00 02 00 00 00 00 00                         ........

L$RTMTIMEBOMB_1320153D-8DA3-4e8e-B27B-0D888223A588

0000   00 C6 76 0A F3 48 CB 01                         ..v..H..

_SC_Alerter


SAC

0000   0F 00 00 00 02 00 00 00 88 00 00 00 14 4C 87 BA    .............L..
0010   FD A5 06 48 6C C7 2A 7A DA 18 FA D6 16 1C DB 3F    ...Hl.*z.......?
0020   CF 1A 90 46 6F 7A ED 6F 0E F6 3F 8F 31 68 E5 B3    ...Foz.o..?.1h..
0030   1D 19 5F 13 7C 23 01 B4 DF 13 6D 71 51 EB F5 A5    .._.|#....mqQ...
0040   AA DB 16 4F 39 A7 37 AC BD 8D EB A8 E1 C2 4B C8    ...O9.7.......K.
0050   1E 7C 88 A5 EF DB 6D B8 4B 1D 4B 6A AD B4 C5 51    .|....m.K.Kj...Q
0060   48 70 92 23 58 28 68 5F 8D 22 26 AD 3A 76 64 EC    Hp.#X(h_."&.:vd.
0070   84 60 7E 00 24 48 90 B9 41 77 28 65 32 7B 81 4C    .`~.$H..Aw(e2{.L
0080   D3 68 01 F3 CA DE 80 56 D9 CC DD 77 70 6D 72 AA    .h.....V...wpmr.
0090   7A 58 CD BA 50 00 00 00 1B EC 37 C0 C8 6E 93 90    zX..P.....7..n..
00A0   CD 77 A0 AE 9C 83 9E C8 06 41 60 57 AC 30 67 63    .w.......A`W.0gc
00B0   10 09 A5 B1 73 EB D9 52 F9 BF D8 D7 A0 22 AC C6    ....s..R.....".."
00C0   5C 1F 9F 94 37 A6 E4 65 27 93 3C 53 00 1E 2A D2    ....7..e'.<S..*.
00D0   AE 80 A0 C2 B9 1D 07 8B 92 19 BA 84 4B E1 C3 FD    ............K...
00E0   6D 47 BC 03 74 E1 6F 8C                         mG..t.o.

_SC_RpcLocator


0083343a-f925-4ed7-b1d6-d95d17a0b57b-RemoteDesktopHelpAssistantAccount
```
15

```
python vol.py -f ../../dumps-memory/dump xp dom1.dmp -o 0xe1490b60 hivedump >SAM.txt
Volatile Systems Volatility Framework 2.0

cat SAM.txt
Last Written         Key
2010-05-03 11:20:49  \SAM
2010-05-03 11:20:49  \SAM\SAM
2010-05-03 11:20:49  \SAM\SAM\Domains
2010-11-26 08:24:47  \SAM\SAM\Domains\Account
2010-05-03 09:49:40  \SAM\SAM\Domains\Account\Aliases
2010-05-03 09:49:53  \SAM\SAM\Domains\Account\Aliases\000003E9
2010-05-03 09:49:53  \SAM\SAM\Domains\Account\Aliases\Members
2010-05-03 09:49:53  \SAM\SAM\Domains\Account\Aliases\Members\S-1-5-21-1229272821-1364589140-1606980848
2010-05-03 09:49:53  \SAM\SAM\Domains\Account\Aliases\Members\S-1-5-21-1229272821-1364589140-1606980848\000003EA
2010-05-03 09:49:40  \SAM\SAM\Domains\Account\Aliases\Names
2010-05-03 09:49:40  \SAM\SAM\Domains\Account\Aliases\Names\HelpServicesGroup
2010-05-03 11:20:49  \SAM\SAM\Domains\Account\Groups
2010-11-26 08:24:46  \SAM\SAM\Domains\Account\Groups\00000201
2010-05-03 11:20:49  \SAM\SAM\Domains\Account\Groups\Names
2010-05-03 11:20:49  \SAM\SAM\Domains\Account\Groups\Names\Aucun
2010-11-26 08:24:46  \SAM\SAM\Domains\Account\Users
```
16

```
python vol.py -f ../../dumps-memory/dump xp dom1.dmp connections
Volatile Systems Volatility Framework 2.0
 Offset(V)  Local Address             Remote Address            Pid
---------- ------------------------ ------------------------ ------
0x8130a698 127.0.0.1:5152           127.0.0.1:1215             2044
0x81299558 192.168.23.89:1251       92.123.3.235:80            1072
```
17

```
python vol.py -f ../../dumps-memory/dump xp dom1.dmp connscan
Volatile Systems Volatility Framework 2.0
 Offset     Local Address             Remote Address            Pid
---------- ------------------------ ------------------------ ------
0x01350680 192.168.23.89:1239       193.51.224.14:80           1964
0x01354d68 192.168.23.89:1219       195.221.189.248:3128       1416
0x01355b48 127.0.0.1:1146           127.0.0.1:12080            3488
0x0162d4c0 192.168.23.89:1115       195.221.189.248:3128       1416
0x01654e68 19.0.0.0:8833            8.44.245.129:51287        16777235
0x01665cb0 127.0.0.1:12080          127.0.0.1:1228             1416
0x01666280 127.0.0.1:12080          127.0.0.1:1220             1416
0x01669d38 127.0.0.1:12080          127.0.0.1:1146             1416
0x01699558 192.168.23.89:1251       92.123.3.235:80            1072
0x0169e9b8 192.168.23.89:1249       75.125.246.130:80          1416
0x016cb948 0.0.0.0:0                0.0.0.0:0                2167191904
0x0170a698 127.0.0.1:5152           127.0.0.1:1215             2044
0x01754e68 0.0.0.0:0                232.164.55.129:0         2167754368
0x01758e68 192.168.23.89:1149       195.221.189.248:3128       1416
0x01778688 127.0.0.1:1148           127.0.0.1:12080            3488
0x0179ae68 127.0.0.1:12080          127.0.0.1:1225             1416
0x017c1008 127.0.0.1:12080          127.0.0.1:1221             1416
0x01a1fdc0 192.168.23.89:1156       195.221.189.248:3128       1416
0x01a34950 192.168.23.89:1212       92.123.3.235:80            1072
0x01a36be0 127.0.0.1:1221           127.0.0.1:12080            3488
0x01ac3e68 0.0.0.0:0                0.0.0.0:0                2171354752
0x01b293d0 192.168.23.89:1214       195.221.189.248:3128       1416
```
18

rité unique) est une valeur unique de longueur variable qui est utilisée pour identifier une entité de sécurité ou d'un groupe de sécurité dans les systèmes d'exploitation Windows. En se référant au « Well-known security identifiers in Windows operating systems », on peut identifier facilement chaque SID révélé.

```
python vol.py -f <chemin du dump> getsids
```
[Fig.20]

Le plugin dlllist permet de lister ou toutes les dll chargées en mémoires ou celles spécifiques à un processus.

```
python vol.py -f <chemin du dump> dlllist -p <PID>
```
[Fig.21]

Il est possible également, à l'aide du plugin memdump de dumper l'espace mémoire adressé par un processus pour l'analyser ensuite avec des outils de reverse engeenering. Reprenons l'écran où nous avons utilisé le plugin psscan qui nous donne l'offset mémoire et le pid de chaque processus et lançons la commande suivante :

```
python vol.py -f <chemin du dump> memdump -o <offset processus> -p <PID>
```
[Fig.22]

On peut par exemple appliquer la commande strings sur le dump réalisé, ici « 2876.dmp » [Fig.23]. On peut également afficher les modules chargés dans le noyau à l'aide du plugin « modules » :

```
python vol.py -f <chemin du dump> modules
```
[Fig.24]

En regardant de près, on s'aperçoit que deux modules intéressants sont chargés en mémoire. Le premier est ioperm.sys et le deuxième win32dd.sys respectivement de cmospwd et de la suite moonsol. Le plugin memdump permet de dumper le module chargé en noyau pour une analyse ultérieure :

```
python vol.py -f <chemin du dump> moddump -o <adresse base> -D <répertoire où écrire le dump>
```

Le nom du dump sera de la forme driver.<adresse hexa>.sys [Fig.25]

19
```
python vol.py -f ../../dumps-memory/dump_xp_dom1.dmp sockets
Volatile Systems Volatility Framework 2.0
 Offset(V)  PID    Port   Proto    Address         Create Time
---------- ------ ------ -------- --------------- -------------------
0x817bf418    232   1252     17 UDP    0.0.0.0         2011-03-30 14:13:44
0x812ac008   1072   1251      6 TCP    0.0.0.0         2011-03-30 14:13:42
0x8124f6a8   1356   1900     17 UDP    192.168.23.89   2011-03-30 14:04:02
0x8124f6a8   1416  12143      6 TCP    127.0.0.1       2011-03-30 14:02:53
0x8170f1b8   1072   1250     17 UDP    127.0.0.1       2011-03-30 14:13:42
0x81620a28      4    445      6 TCP    0.0.0.0         2011-03-30 14:02:42
0x81ad1478   1416  12465      6 TCP    127.0.0.1       2011-03-30 14:02:53
0x81ad1478    984    135      6 TCP    0.0.0.0         2011-03-30 14:02:48
0x81b551c8      4    139      6 TCP    192.168.23.89   2011-03-30 14:02:42
0x81e9a6b8    232   1346     17 UDP    0.0.0.0         2011-03-30 14:03:19
0x81b2b458   1416  12993      6 TCP    127.0.0.1       2011-03-30 14:02:54
0x81b2b458      4    137     17 UDP    192.168.23.89   2011-03-30 14:02:42
0x8126c6b0   1072    123     17 UDP    127.0.0.1       2011-03-30 14:03:30
0x81b29958   1416  12563      6 TCP    127.0.0.1       2011-03-30 14:02:54
0x81b2c278   1416  12110      6 TCP    127.0.0.1       2011-03-30 14:02:53
0x8127fd68   1288   1030      6 TCP    127.0.0.1       2011-03-30 14:03:24
0x8127fd68      4    138     17 UDP    192.168.23.89   2011-03-30 14:02:42
0x81bac200   2204   1070     17 UDP    127.0.0.1       2011-03-30 14:06:42
0x81c41470   1416  12025      6 TCP    127.0.0.1       2011-03-30 14:02:53
0x81323268   1356   1900     17 UDP    127.0.0.1       2011-03-30 14:04:02
0x81323268   1072    123     17 UDP    192.168.23.89   2011-03-30 14:03:30
0x8128be98   2044   5152      6 TCP    127.0.0.1       2011-03-30 14:03:19
0x8128be98      4    445     17 UDP    0.0.0.0         2011-03-30 14:02:42
0x81620b98   1416  12080      6 TCP    127.0.0.1       2011-03-30 14:02:53
0x81b4f450   1416  12995      6 TCP    127.0.0.1       2011-03-30 14:02:53
0x81ad1270   1416  12119      6 TCP    127.0.0.1       2011-03-30 14:02:53
```

20
```
python vol.py -f ../../dumps-memory/dump_xp_dom1.dmp getsids
Volatile Systems Volatility Framework 2.0
System (4): S-1-5-18 (Local System)
System (4): S-1-5-32-544 (Administrators)
System (4): S-1-1-0 (Everyone)
System (4): S-1-5-11 (Authenticated Users)
smss.exe (596): S-1-5-18 (Local System)
smss.exe (596): S-1-5-32-544 (Administrators)
smss.exe (596): S-1-1-0 (Everyone)
smss.exe (596): S-1-5-11 (Authenticated Users)
csrss.exe (664): S-1-5-18 (Local System)
csrss.exe (664): S-1-5-32-544 (Administrators)
csrss.exe (664): S-1-1-0 (Everyone)
csrss.exe (664): S-1-5-11 (Authenticated Users)
winlogon.exe (688): S-1-5-18 (Local System)
winlogon.exe (688): S-1-5-32-544 (Administrators)
winlogon.exe (688): S-1-1-0 (Everyone)
winlogon.exe (688): S-1-5-11 (Authenticated Users)
services.exe (732): S-1-5-18 (Local System)
services.exe (732): S-1-5-32-544 (Administrators)
services.exe (732): S-1-1-0 (Everyone)
services.exe (732): S-1-5-11 (Authenticated Users)
lsass.exe (744): S-1-5-18 (Local System)
lsass.exe (744): S-1-5-32-544 (Administrators)
lsass.exe (744): S-1-1-0 (Everyone)
```

De la même façon qu'un processus peut être caché pour être discret, un module peut être caché. Le plugin modscan va nous aider à afficher un éventuel module caché :

```
python vol.py -f <chemin du dump> modscan > <chemin et nom du fichier.txt à créer>
```

Le nom du dump sera de la forme driver.<adresse hexa>.sys

CONCLUSION

Nous venons de voir comment récupérer des informations importantes en mémoire après un Dump de celle-ci. Comme dit plus haut, il faut avoir d'abord avoir compromis la machine (être devenu administrateur) via des méthodes de hacking différentes telles que les exploits, les failles firewire, un boot avec des outils spécialisés nous permettant d'être administrateur. Ce travail de forensic est utilisé par exemple lors de saisies d'ordinateurs pour trouver des preuves contre le criminel.

Vous pouvez retrouver mes vidéos sur Python, Scapy, le forensic en Python sur ma chaîne youtube : http://www.youtube.com/c/franckebel

21
```
python vol.py -f ../../dumps-memory/dump_xp_dom1.dmp dlllist -p 2288
Volatile Systems Volatility Framework 2.0
************************************************************************
hkcmd.exe pid:   2288
Command line : "C:\WINDOWS\system32\hkcmd.exe"
Service Pack 3

Base             Size             Path
0x00400000       0x013000         C:\WINDOWS\system32\hkcmd.exe
0x7c910000       0x0b9000         C:\WINDOWS\system32\ntdll.dll
0x7c800000       0x106000         C:\WINDOWS\system32\kernel32.dll
0x64d00000       0x031000         C:\Program Files\Alwil Software\Avast5\snxhk.dll
0x7e390000       0x091000         C:\WINDOWS\system32\USER32.dll
0x77ef0000       0x049000         C:\WINDOWS\system32\GDI32.dll
0x77da0000       0x0ac000         C:\WINDOWS\system32\ADVAPI32.dll
0x77e50000       0x092000         C:\WINDOWS\system32\RPCRT4.dll
0x77fc0000       0x011000         C:\WINDOWS\system32\Secur32.dll
0x774a0000       0x13d000         C:\WINDOWS\system32\ole32.dll
0x77be0000       0x058000         C:\WINDOWS\system32\msvcrt.dll
0x770e0000       0x08b000         C:\WINDOWS\system32\OLEAUT32.dll
0x10000000       0x013000         C:\WINDOWS\system32\hccutils.DLL
0x76320000       0x01d000         C:\WINDOWS\system32\IMM32.DLL
0x5b090000       0x038000         C:\WINDOWS\system32\uxtheme.dll
0x76f80000       0x07f000         C:\WINDOWS\system32\CLBCATQ.DLL
0x77000000       0x0d4000         C:\WINDOWS\system32\COMRes.dll
0x77bd0000       0x008000         C:\WINDOWS\system32\VERSION.dll
0x00970000       0x2da000         C:\WINDOWS\system32\xpsp2res.dll
0x00f50000       0x00e000         C:\WINDOWS\system32\igfxsrvc.dll
0x74690000       0x04c000         C:\WINDOWS\system32\MSCTF.dll
0x75140000       0x02e000         C:\WINDOWS\system32\msctfime.ime
0x00fc0000       0x024000         C:\WINDOWS\system32\igfxres.dll
```

22
```
python vol.py -f ../../dumps-memory/dump_xp_dom1.dmp  -o 0x0162fbd0 -p 2876 memdump -D test
Volatile Systems Volatility Framework 2.0
************************************************************************
Writing unsecapp.exe [  2876] to 2876.dmp
```

23
```
strings test/2876.dmp | more
rs commu
omSpec=C
_HOST_CH
cuments c
inistrateur
LOGONSERVER=\\IUT_270_MODELE
NUMBER_OF_PROCESSORS=1
OS=Windows_NT
Path=C:\Python27\;C:\Python27\Scripts;C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem;C:\D
ocuments and Settings\prof\Application Data\Python\Scripts
PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.py;.pyw
PROCESSOR_ARCHITECTURE=x86
PROCESSOR_IDENTIFIER=x86 Family 15 Model 2 Stepping 9, GenuineIntel
PROCESSOR_LEVEL=15
PROCESSOR_REVISION=0209
```

24
```
$ python vol.py -f ../../dumps-memory/dump_xp_dom1.dmp modules
Volatile Systems Volatility Framework 2.0
Offset(V)  File                                           Base       Size     Name
0x81ffc3a8 \WINDOWS\system32\ntoskrnl.exe                 0x00804d7000 0x217280 ntoskrnl.exe
0x81ffc340 \WINDOWS\system32\hal.dll                      0x00806ef000 0x020300 hal.dll
0x81ffc2d8 \WINDOWS\system32\KDCOM.DLL                    0x00f8a37000 0x002000 kdcom.dll
0x81ffc268 \WINDOWS\system32\BOOTVID.dll                  0x00f8947000 0x003000 BOOTVID.dll
0x81ffc200 ACPI.sys                                       0x00f84e7000 0x02f000 ACPI.sys
0x81ffc190 \WINDOWS\system32\DRIVERS\WMILIB.SYS           0x00f8a39000 0x002000 WMILIB.SYS
0x81ffc128 pci.sys                                        0x00f84d6000 0x011000 pci.sys
0x81ffc0b8 isapnp.sys                                     0x00f8537000 0x00a000 isapnp.sys
0x81ffc048 pciide.sys                                     0x00f8aff000 0x001000 pciide.sys
0x81fef008 \WINDOWS\system32\DRIVERS\PCIIDEX.SYS          0x00f87b7000 0x007000 PCIIDEX.SYS
0x81feff98 intelide.sys                                   0x00f8a3b000 0x002000 intelide.sys
0x81feff28 MountMgr.sys                                   0x00f8547000 0x00b000 MountMgr.sys
0x81fefeb8 ftdisk.sys                                     0x00f84b7000 0x01f000 ftdisk.sys
0x81fefe48 dmload.sys                                     0x00f8a3d000 0x002000 dmload.sys
0x81fefde0 dmio.sys                                       0x00f8491000 0x026000 dmio.sys
0x81fefd70 PartMgr.sys                                    0x00f87bf000 0x005000 PartMgr.sys
0x81fefd00 pavboot.sys                                    0x00f87c7000 0x006000 pavboot.sys
0x81fefc90 VolSnap.sys                                    0x00f8557000 0x00e000 VolSnap.sys
0x81fefc28 atapi.sys                                      0x00f8479000 0x018000 atapi.sys
0x81fefbc0 disk.sys                                       0x00f8567000 0x009000 disk.sys
0x81fefb50 \WINDOWS\system32\DRIVERS\CLASSPNP.SYS         0x00f8577000 0x00d000 CLASSPNP.SYS
0x81fefae0 fltMgr.sys                                     0x00f8459000 0x020000 fltMgr.sys
0x81fefa70 KSecDD.sys                                     0x00f8442000 0x017000 KSecDD.sys
0x81fefa08 Ntfs.sys                                       0x00f83b5000 0x08d000 Ntfs.sys
0x81fef9a0 NDIS.sys                                       0x00f8388000 0x02d000 NDIS.sys
0x81fef938 Mup.sys                                        0x00f836e000 0x01a000 Mup.sys
0x81fd34c0 \SystemRoot\system32\DRIVERS\intelppm.sys       0x00f8677000 0x00a000 intelppm.sys
```

25
```
python vol.py -f ../../dumps-memory/dump_xp_dom1.dmp moddump -o 0x00eeb49000 -u -D test
Volatile Systems Volatility Framework 2.0
Dumping ioperm.sys, Base: eeb49000 output: driver.eeb49000.sys

ls test
2876.dmp  driver.ee5be000.sys  driver.eeb49000.sys
```

Quel développeur full-stack êtes-vous ?

Une publication Nefer-IT, 7 avenue Roger Chambonnet, 91220 Brétigny sur Orge - redaction@programmez.com
Tél. : 01 60 85 39 96 - Directeur de la publication & Rédacteur en chef : François Tonic
Secrétaire de rédaction : Olivier Pavie
Nos experts techniques : S. Noirpoudre, S. Abélard, S. Gombaud, C. Outreville, W. Chegham, D. Duplan, M. Luce-Lucas, M. Lecordonnier, A. Rochedy, E. Vernié, Y. Skrzypczyk, P. Lamarche, C. Pichaud, C. Chevalier, F. Sagez, N. Baptiste, J. Antoine, M. Frappat, T. Leriche-Dessirier, S. Berberat, P-A Sunier, A. Pacaud, F. Ebel

Couverture : © Tribalium © Angular - **Maquette :** Pierre Sandré.
Publicité : PC Presse, Tél.: 01 74 70 16 30, Fax : 01 40 90 70 81 - pub@programmez.com.
Imprimeur : S.A. Corelio Nevada Printing, 30 allée de la recherche, 1070 Bruxelles, Belgique.
Marketing et promotion des ventes : Agence BOCONSEIL - Analyse Media Etude - Directeur : Otto BORSCHA oborscha@boconseilame.fr
Responsable titre : Terry MATTARD Téléphone : 09 67 32 09 34

Contacts : Rédacteur en chef : ftonic@programmez.com - Rédaction : redaction@programmez.com - Webmaster : webmaster@programmez.com - **Publicité :** pub@programmez.com - Evenements / agenda : redaction@programmez.com

Dépôt légal : à parution - Commission paritaire : 1220K78366 - ISSN : 1627-0908 - © NEFER-IT / Programmez, janvier 2017

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

Sur abonnement ou en kiosque

Le magazine des pros de l'IT

Mais aussi sur le web

Ou encore sur votre tablette

www.linformaticien.com

L'INFORMATICIEN

[2017]

DÉVELOPPEZ 10 FOIS PLUS VITE

WINDEV®

WINDEV 22: ATELIER DE DÉVELOPPEMENT PROFESSIONNEL, COMPLET EN STANDARD

Gestion du cycle de vie complet: Idée, Conception, Développement, Génération, Déploiement, Exploitation • Un code multi-plateformes Windows, Linux, Java, Internet, Mobiles • Environnement ALM complet • Toutes les bases de données sont supportées, Big Data • Inclus: HFSQL, base de données locale, Client/Serveur, cluster, embarquée et cloud • Puissant RAD • Intégration continue • Tableau de bord de vos applications • Audit statique & dynamique • Générateur de fenêtres (UI) visuel & intuitif • Utilisation facile de charte graphique • Héritage et surcharge d'interface • Tous les champs (contrôles) sont très puissants et livrés en standard: Champ de saisie, Tableau de bord, Champ Table, Champ Graphe, etc • FAA: chaque application bénéficie automatiquement de Fonctionnalités Automatiques: export vers Excel, vers Word, envoi d'email, etc • Sécurité : Mot de passe de vos applications • Puissant générateur de rapports et codes-barres • Langage de 5ème génération: WLangage • Editeur de code intuitif avec puissant débogueur • Tests unitaires et tests automatiques • Versioning (GDS/SCM) • Webservices SOAP et Rest • Modélisation Merise et UML • .NET, 3-Tier, MVP • Support de tous les standards: XML, USB, Bluetooth, NFC, J2EE, OLE, ActiveX, RPC, SaaS, SMTP, FTP, OPC, DLNA, IoT, Sockets, API, Webservices... • Lien avec Lotus Notes, SAP, Google, Outlook • Multimédia, Domotique • Livré avec des centaines d'exemples et d'assistants • Génère le Dossier technique d'un clic • Télémétrie pour connaître l'utilisation réelle de vos applications • Générateur d'aide • Support de 64 langues étrangères par application • Générateur de procédures d'installation: local, CD, USB, Internet, Réseau, Push... • Robot de surveillance: surveillez vos applications • Gestion des suggestions et incidents utilisateurs • Support Technique Personnalisé Gratuit* • ...

Consultez plus de 100 Témoignages de professionnels sur le site pcsoft.fr

Elu
«Langage le plus productif du marché»

VERSION EXPRESS GRATUITE
Téléchargez-la !

922 NOUVEAUTÉS

Tél Paris: 01 48 01 48 88 *Tél Montpellier: 04 67 032 032*

Plus de 100 témoignages **WWW.PCSOFT.FR**

Logiciel professionnel. * 15 requêtes gratuites sur la version en cours de commercialisation