PROGRAMMEZ.COM

PROGRAMMEZ!

Mensuel n°177 - Septembre 2014

le magazine du développeur

Comment CHOISIR son LANGAGE de PROGRAMMATION ?

L'apprentissage du code à l'école : une bonne idée ?

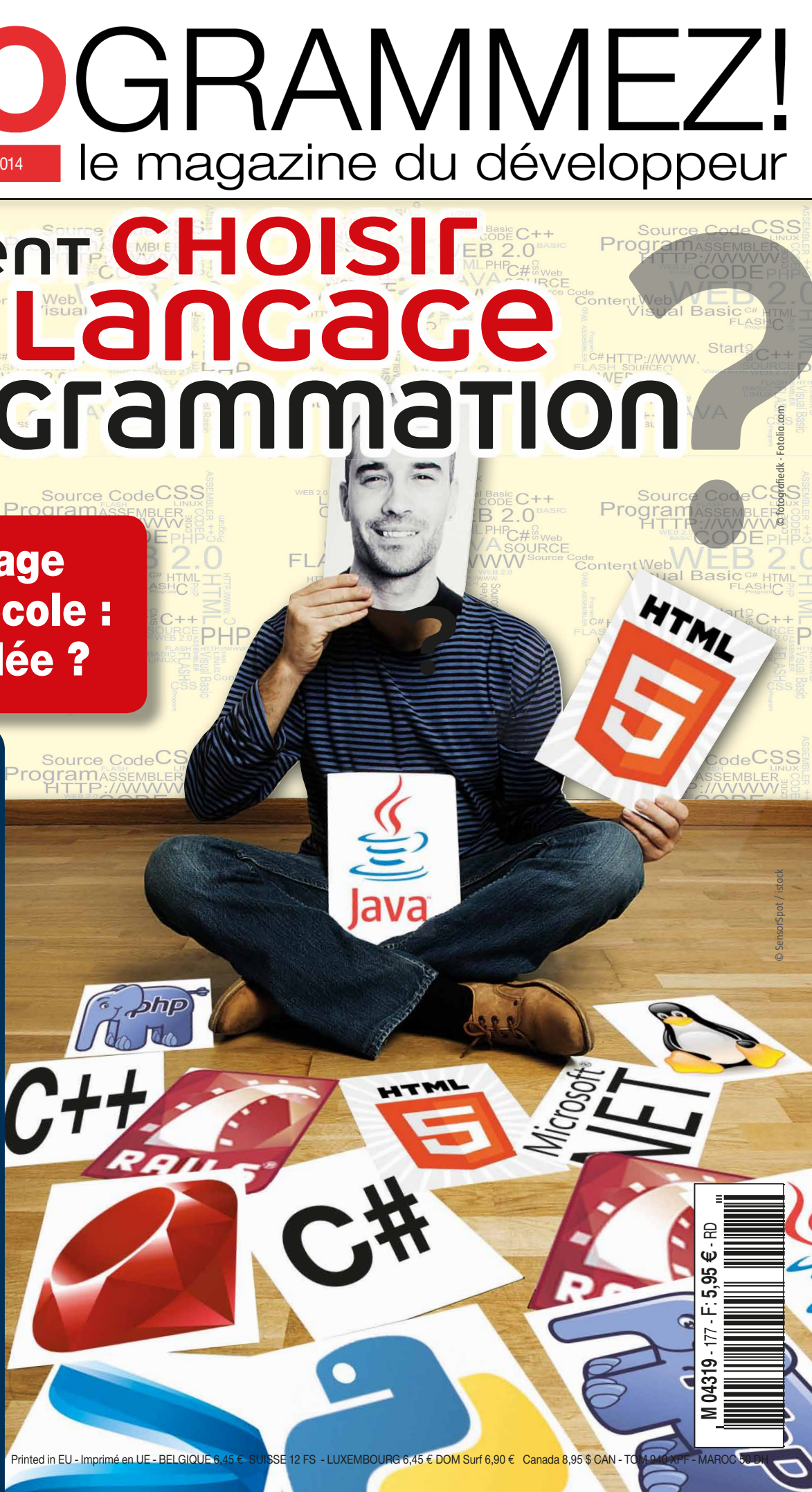
- **PHP**
  - le futur du langage
  - Symfony
  - Zend Framework
  - PHP dans le Cloud
- **Babylon.JS** le moteur 3D détonnant

Le **futur** d'**Android**

Optimiser la **JVM**

Introduction à **AngularJS**

- **Antiquité** L'aventure du MSX



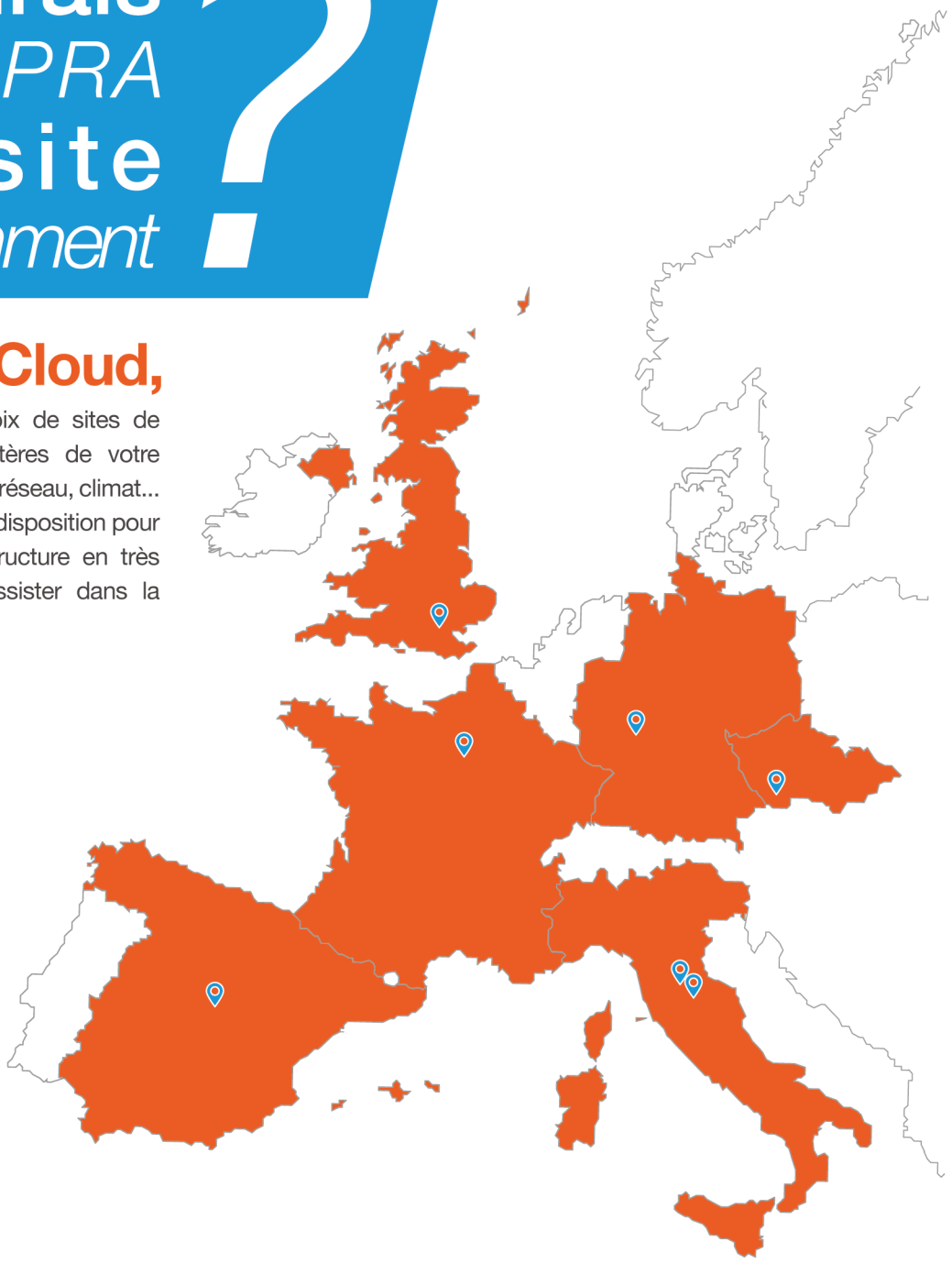
M 04319 - 177 - F: 5,95 € - RD

Printed in EU - Imprimé en UE - BELGIQUE 6,45 € SUISSE 12 FS - LUXEMBOURG 6,45 € DOM Surf 6,90 € Canada 8,95 $ CAN - TOM 940 XPF - MAROC 50 DH

Je voudrais *bâtir un PRA* **multi-site** *Je fais comment* ?

Avec Aruba Cloud,

vous disposez d'un large choix de sites de secours, en fonction des critères de votre stratégie de sécurité: proximité, réseau, climat... Nos équipes sont aussi à votre disposition pour vous aider à bâtir une infrastructure en très haute disponibilité et vous assister dans la définition de votre stratégie.



- 3 hyperviseurs
- 6 datacenters en Europe
- APIs et connecteurs
- 70+ templates
- Contrôle des coûts

Nous avons choisi Aruba Cloud car nous bénéficions d'un haut niveau de performance, à des coûts contrôlés et surtout car ils sont à dimension humaine, comme nous. Xavier Dufour - Directeur R&D - ITMP

Contactez-nous! 0810 710 300 www.arubacloud.fr

aruba CLOUD

Cloud Public | Cloud Privé | Cloud Hybride | Cloud Storage | Infogérance

MY COUNTRY. MY CLOUD.*

*MON PAYS, MON CLOUD.

édito

Sortez vos claviers. Aujourd'hui {

On ne pouvait pas avoir meilleur timing. Nous vous annoncions pour la rentrée un dossier sur la découverte de la programmation aux enfants. Eh paf ! En plein été, le gouvernement français annonce que la programmation sera à l'école primaire dès la rentrée 2014, sous la forme d'une initiation ! Pour le moment, elle sera facultative et réalisée en dehors des heures de cours. Les associations sont appelées à l'aide pour structurer cette initiation... Mais cela pose plusieurs questions : quel contenu pour cette initiation, qui fera le cours, comment seront formés les professeurs qui vont animer ce cours, quels matériels, quels logiciels... L'ambition est louable, mais la réalisation sera-t-elle à la hauteur ?

Autre dossier chaud pour cette rentrée 2014, PHP. Pourquoi chaud ? Car la communauté bouillonne. Nous ferons le point de la situation sur les derniers outils, les nouveautés attendues dans le langage. Le futur de PHP s'écrira avec la version 7, la version 6 n'existera pas, pour faire oublier PHP 6.

Autre gros dossier de notre rentrée 2014, une problématique anodine, mais si importante pour le développeur (débutant et confirmé) : comment (bien) choisir ses langages de programmation ? Finalement, je me demande si la majorité des développeurs se pose réellement la question. Car nous sommes toutes et tous un peu suiveurs. HTML5 est à la mode, et tout le monde s'y met. Rails c'est top, et hop, on le met sur notre liste. Aujourd'hui, le développeur doit absolument être polyglotte et maîtriser plusieurs langages pour être un véritable caméléon et être capable de passer d'une plate-forme à une autre immédiatement. C'est aussi la capacité du développeur à apprendre un nouveau langage très rapidement. Se dédier à une plate-forme, un langage, c'est le risque d'enfermer sa carrière et de rater des opportunités. Programmez ! vous propose de vous poser les bonnes questions et de trouver les critères de choix...

Dans ce numéro de rentrée, nous vous proposons du très lourd : F#, optimiser la machine virtuelle Java, SVG, AngularJS, BabylonJS, comment optimiser une scène 3D, Google I/O et le futur d'Android, sans oublier notre Time Machine du mois : MSX ! Et nous ferons aussi le point sur la Maker Faire Paris (utilisez le QRCode pour accéder à l'article).

Enjoy !

François Tonic
Directeur de la publication & rédacteur en chef
ftonic@programmez.com

sommaire

À LIRE DANS LE PROCHAIN NUMÉRO
n° 178 en kiosque le 27 septembre 2014

DRUPAL AVANCÉ
Allons plus loin avec Drupal.

CLOUD COMPUTING
Utilisez les conteneurs Docker.

LANGAGE
Comment utiliser les lambdas en Javascript ?

CODING4FUN
Raspberry Pi + NFC = ?

Respect de la vie privée et de la confidentialité des données

63% *des répondants français affirment ne pas vouloir sacrifier la confidentialité de leurs données au profit de plus de simplicité et de confort d'utilisation.*

39% *passent entre 3 et 4 heures en ligne par jour...*

1 français sur 2 *n'a pas confiance dans les compétences des entreprises pour protéger sa vie privée,*

3 français sur 4 *ne font pas confiance en l'éthique des entreprises*

88% *affirment ne pas apprécier que quelqu'un possède des informations sur eux ou leurs habitudes en ligne*

59% *des répondants estiment avoir moins de vie privée en ligne qu'il y a un an*

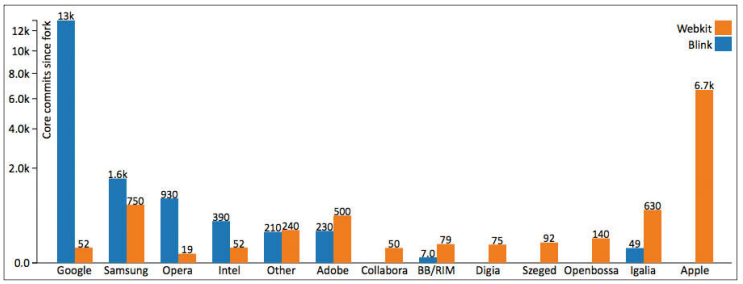
Près de 70% *des Français estiment que le gouvernement n'en fait pas assez pour protéger leur confidentialité*

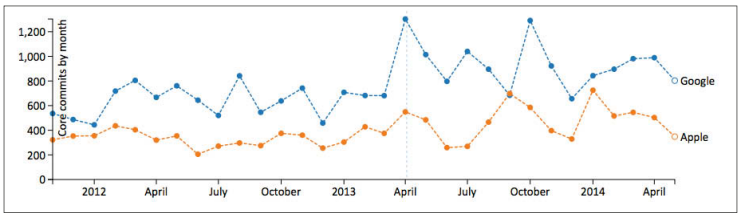
91% *estiment qu'il devrait y avoir des lois pour interdire la vente et/ou l'achat de leurs données sans leur consentement.*

(source : étude Privacy Index, EMC, juin 2014)

1 an de Blink et de Webkit

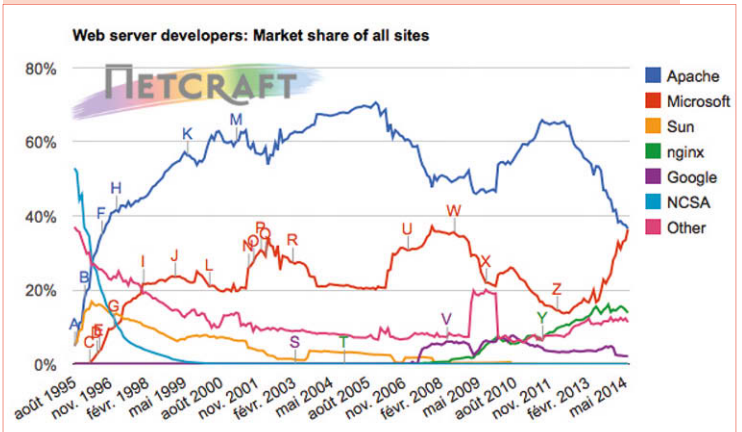
Philip Rogers a publié un très intéressant post sur les évolutions des moteurs WebKit et Blink depuis 1 an. Pour mémoire, Blink est un fork de WebKit initié par Google (printemps 2013). WebKit s'était du lourd : 1,8 million de lignes de code en C++, 2500 commits par mois et leader du web mobile. Les contributions à Blink ont rapidement explosé, dépassant WebKit et WebKit Core. Cela s'explique par la phase de lancement de Blink. Les deux projets ont réduit les lignes de code (bonne nouvelle) même si WebKit reste encore très gros (environ 800 000 lignes). Sans surprise, Google est le principal contributeur de Blink (étonnant non ?), puis suivent Samsung, Opera et Intel. Côté WebKit, Apple reste le principal contributeur, mais plusieurs éditeurs travaillent sur les deux moteurs. Source : http://browserg.nom.es

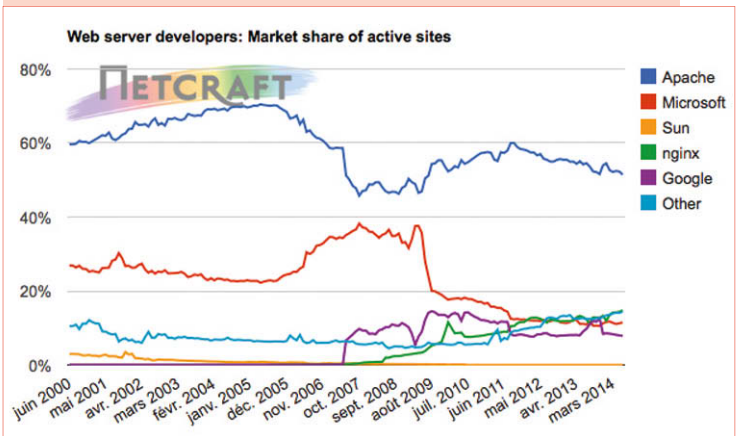




Web Server Survey : juin 2014

Chaque mois, Netcraft dévoile son rapport sur les serveurs web en analysant presque 1 milliard de sites web. Plus que jamais, le marché se partage entre Apache et Microsoft. Ce dernier semble très en forme : Apache reste leader du marché, mais IIS revient très fort. Le marché s'équilibre, 37 % pour Apache, 33 % pour Microsoft. Les autres serveurs web sont très loin derrière : nginx se contente de 14,6 %. Cependant, attention, sur les sites réellement actifs, Apache est largement devant (52 %), Microsoft seulement 3e.
Source : http://www.netcraft.com





MIROIR, MIROIR, QUEL EST LE LANGAGE LE PLUS UTILISÉ ?

L'index TIOBE donne chaque mois les langages les plus utilisés par les développeurs. Il s'agit d'un indicateur basé sur les recherches web. Des % à manipuler avec précaution.

Août 2014	Août 2013	Tendances	Langage	%	Evolution (en %)
1	2	↑	C	16,401	+0,43
2	1	↓	Java	14,984	-0,99
3	4	↑	Objective-C	9,552	+1,47
4	3	↓	C+ +	4,695	-4,68
5	7	↑	Basic	3,635	-0,24
6	6	=	C#	3,409	-2,71
7	8	↑	Python	3,121	-0,48
8	5	↓	PHP	2,864	-3,83
9	11	↑	Perl	2,218	+0,18
10	9	↓	JavaScript	2,172	+0,08

Pas de changement pour le trio de tête : C, Java, Objective-C.
Tous les langages, ou presque, subissent des baisses. Le marché est très segmenté. Swift, le nouveau langage d'Apple, était classé 23e après une montée dans les 20 premiers.

Plans :

Apple modifie chaque jour ses cartes

Bracelet connecté Microsoft ?

En attendant l'hypothétique montre Apple, Microsoft pourrait suivre le mouvement dès cet automne...

Tesla :

des brevets ouverts et gratuits !
La révolution de la voiture 2.0 est en marche

Bitcoin

devient légal en Californie

MICROSOFT FAIT LE MÉNAGE DANS IE

Comment faire disparaître les versions anciennes d'Internet Explorer ? En arrêtant de les supporter et en forçant la migration si les utilisateurs et entreprises veulent toujours bénéficier de la maintenance... Ainsi, une version minimale sera imposée sur Windows Vista, 7, 8.1 et Server... à partir du 12 janvier 2016.

Windows Platform	Internet Explorer Version
Windows Vista SP2	Internet Explorer 9
Windows Server 2008 SP2	Internet Explorer 9
Windows 7 SP1	Internet Explorer 11
Windows Server 2008 R2 SP1	Internet Explorer 11
Windows 8.1	Internet Explorer 11
Windows Server 2012	Internet Explorer 10
Windows Server 2012 R2	Internet Explorer 11

Chrome passe en 64 bits

Hyperlapses :

Microsoft Research travaille sur un algorithme pour stabiliser les vidéos.

4.5 :

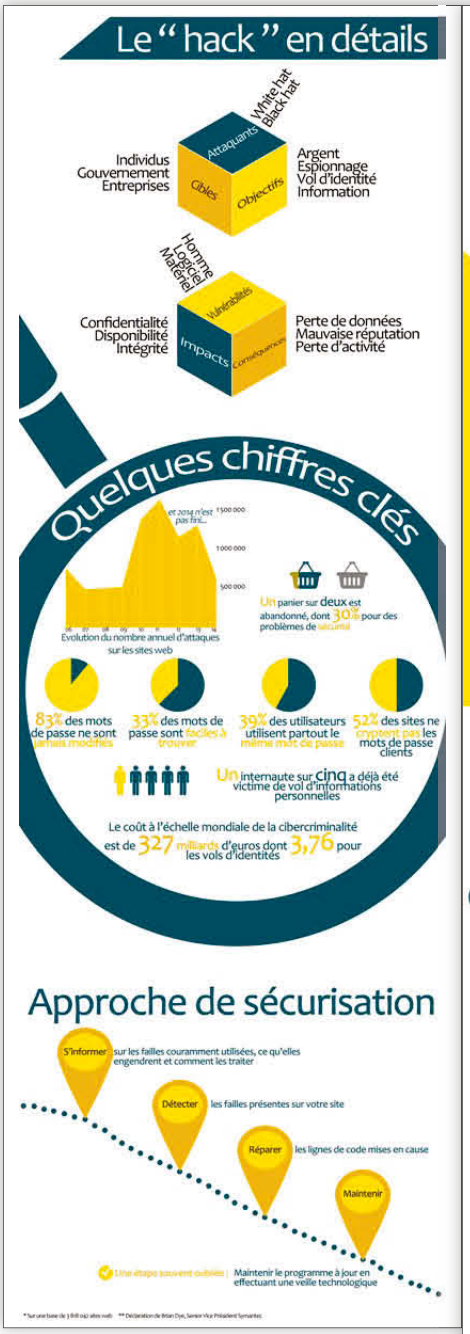
la nouvelle version d'Open GL

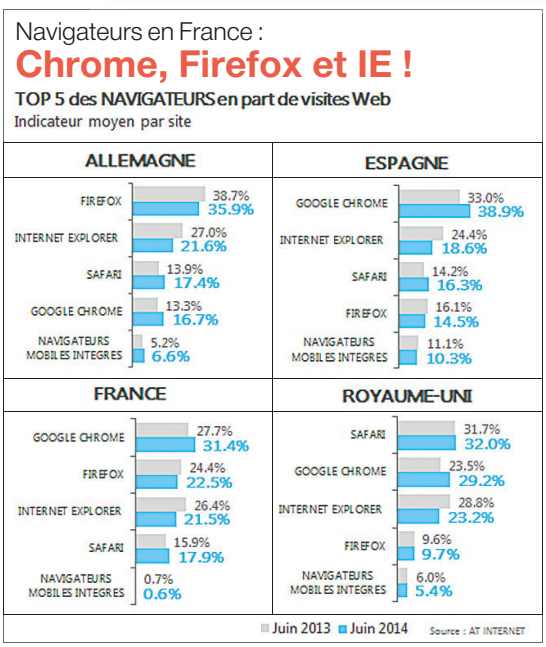
VIV :

les créateurs de Siri (Apple) veulent aller beaucoup plus loin avec leur nouvel assistant virtuel, Viv.

Nokia X :

Microsoft change de stratégie et n'en veut plus mais lance un mobile à 19 € !





Google I/O 2014 : Android et le monde des objets connectés à l'honneur !

Depuis maintenant 7 ans, la conférence Google I/O est l'occasion pour Google de communiquer sur ses produits et technologies en fédérant la communauté toujours friande des keynotes où les annonces de nouveautés sont légion.

Cette année encore, les développeurs et le grand public n'auront pas été déçus avec des annonces centrées autour d'Android : nouveau design made in Google nommé Material Design, preview de la future mouture d'Android, sans oublier des SDK Android dédiés au monde des objets connectés. Toutes ces annonces auront mis en exergue la volonté de Google de partir à la conquête du monde des objets connectés qui s'annonce comme le nouveau terrain de bataille des géants de l'informatique. En outre, l'annonce du programme Android One, destiné aux smartphones d'entrée de gamme, confirme que Google part désormais à l'assaut du marché des émergents. Au final, l'ensemble de la conférence aura tourné autour des 3 D : Designer, Développer et Distribuer.

Android L

Pour la première fois depuis le lancement d'Android, Google met une version preview à disposition des développeurs. Attendue pour l'automne, Android L amène de grands changements tant au niveau visuel avec l'arrivée du Material Design qu'au niveau des nouvelles fonctionnalités qu'elle introduit et qui ouvrent la voie aux autres déclinaisons d'Android. Outre l'arrivée du Material Design, dont nous reparlerons par la suite, cette nouvelle mouture voit l'arrivée d'ART comme runtime par défaut. Introduit expérimentalement en version 4.4, ART remplace Dalvik en devenant la machine virtuelle par défaut de l'OS.

Parmi les principaux avantages d'ART, on citera la compilation AOT (Ahead-Of-Time) qui peut améliorer nettement les performances des applications. En effet, à l'installation d'une application, ART va la compiler via l'outil dex2oat et générer une version compilée, exécutable et optimisée pour le périphérique cible. En outre, le garbage collector d'ART propose un fonctionnement amélioré réduisant les temps de latence. La majorité des applications Android existantes devrait fonctionner sans modifications avec ART, mais le développeur devra néanmoins être particulièrement attentif aux possibles incompatibilités si son application s'appuie sur JNI pour exécuter du code C/C++ ou s'il utilise des outils de génération de code non standard tels que des obfuscateurs.

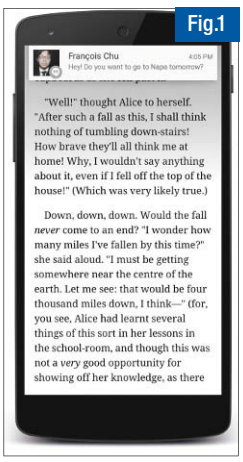


Fig.1

Notifications flottantes

Autre nouveauté Android L, le sérieux lifting subi par les notifications qui peuvent dorénavant apparaître au sein de l'écran de verrouillage d'où elles peuvent être contrôlées et mise à jour sans déverrouillage. D'autre part, les notifications flottantes font leur apparition et permettront d'afficher du contenu à l'utilisateur au sein de fenêtres flottantes managées par l'OS (Fig.1).

La visibilité d'une notification sur l'écran de verrouillage peut être réglée finement via la méthode setVisibility de la classe Notification.Builder. En sus, il est maintenant préférable de laisser le système gérer entièrement les notifications et leur contexte. Ainsi, l'ajout de sons ou de vibration à une notification devra se faire exclusivement via les méthodes setSound et setVibrate du builder précédemment cité. Enjeu crucial des systèmes embarqués, la gestion de la batterie n'est pas oubliée avec l'intégration du projet Volta qui apporte un ensemble d'outils et d'API visant à optimiser l'exécution des applications. Le but étant in fine de prolonger au maximum l'autonomie de la batterie. Pour mieux comprendre les évènements consommateurs d'énergie, l'outil batterystats est ajouté au SDK. La première étape va ainsi être de générer les données statistiques d'utilisation de la batterie sur un périphérique pour une application :

```
adb shell dumpsys batterystats --charged <package-name>
```

Une fois le fichier bugreport.txt généré, il est possible de générer un fichier HTML facilitant la visualisation des évènements liés à la batterie :

```
historian.par [-p powerfile] bugreport.txt > out.html
```

On notera également l'apparition de l'API android.app.job.JobScheduler dont l'objectif est d'optimiser la durée de vie de la batterie en créant des jobs qui seront exécutés de manière asynchrone par le système sous certaines conditions. Il est ainsi concevable de planifier une tâche consommatrice en énergie uniquement lorsque l'appareil est en charge ou bien de planifier une tâche réseau uniquement en mode Wi-Fi. La définition des critères d'exécution d'un job se fait au travers de la classe JobInfo qui expose un certain nombre de méthodes dédiées. On peut par exemple planifier une tâche à exécuter uniquement lorsque l'appareil sous-jacent sera en réseau illimité :

```
JobInfo uploadTask = new JobInfo.Builder(mJobId, mService
Component)
   .setRequiredNetworkCapabilities(JobInfo.NetworkType.UNMETERED)
   .build();

JobScheduler jobScheduler = (JobScheduler) context.getSystem
Service(Context.JOB_SCHEDULER_SERVICE)
jobScheduler.schedule(uploadTask);
```

À côté de ses nouveautés majeures, Android L s'amène avec un support du 64 bits qui bénéficiera de manière transparente à toutes les applications hormis celles exécutant du code natif qui devront logiquement s'adapter. Le mode Bluetooth Low Energy (BLE) est à présent supporté par l'OS. Enfin, une nouvelle API pour la gestion des caméras débarque sous le package android.hardware.camera2 . Plus pratique d'utilisation et centrée autour du helper CameraManager, elle propose des possibilités accrues en termes de traitements d'images.

Pour développer en avant-première sur Android L, il suffit d'aller sur le site suivant : http://developer.android.com/preview/setup-sdk.html . Au programme, tous les détails sur les démarches à suivre pour installer le SDK d'une part et l'image système Android L en preview d'autre part.

Material Design

Introduit il y a près de 3 ans, le thème Holo aura permis d'offrir aux développeurs d'applications Android un modèle visuel commun unifiant les interfaces des smartphones et des tablettes. Pour aller encore plus loin,

Complétez votre collection PROGRAMMEZ! le magazine du développeur

Prix unitaire : 6 € (Frais postaux inclus) **France métropolitaine uniquement.**

nouveau

Tout Programmez! sur une clé USB

Tous les numéros de Programmez! depuis le n°100.

29,90 €*

Clé USB 2 Go.
Testé sur Linux, OS X, Windows. Les magazines sont au format PDF.

* *tarif pour l'Europe uniquement. Pour les autres pays, voir la boutique en ligne*

vous pouvez aussi commander directement sur notre site internet : www.programmez.com

☐ ***169*** : |__| exemplaire(s)
☐ ***170*** : |__| exemplaire(s)
☐ ***171*** : |__| exemplaire(s)
☐ ***172*** : |__| exemplaire(s)
☐ ***174*** : |__| exemplaire(s)
☐ ***175*** : |__| exemplaire(s)
☐ ***176*** : |__| exemplaire(s)

☐ ***La Clé USB avec les numéros de Programmez depuis le n°100***
29,90 € (Tarif pour Europe uniquement)

Commande à envoyer à :
Programmez!
7, avenue Roger Chambonnet
91220 Brétigny sur Orge

soit |__|__| exemplaires x 6 € = |__|__| € + ☐ 29,90 € soit au **TOTAL** = |__|__|__|__|__| €

Prix unitaire : 6 € (Frais postaux inclus), France métropolitaine uniquement.

PROG 176

☐ M. ☐ Mme ☐ Mlle Entreprise : |__|__|__|__|__|__|__|__| Fonction : |__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|

Prénom : |__|__|__|__|__|__|__|__|__|__|__|__|__| Nom : |__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|

Adresse : |__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|

Code postal : |__|__|__|__|__| Ville : |__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|

Tarifs France métropolitaine

Tél : |__|__|__|__|__|__|__|__|__|__| ***(Attention, e-mail indispensable)***

E-mail : |__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__| @ |__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|__|

Règlement par chèque à l'ordre de Programmez !

Valable jusqu'au 27 septembre 2014
Quantités limitées

Google lance Material Design, un véritable langage visuel qui apporte une approche compréhensive en incorporant des principes de design pour les interfaces utilisateurs. Le but premier du Material Design étant de fournir des mouvements fluides sur surfaces tactiles tout en gardant un support fort pour les souris et claviers afin de pouvoir cibler une multitude d'appareils aux écrans hétérogènes.

Le Material Design définit un certain nombre de patterns UI en termes d'animation, style et layout tout en mettant en avant des recommandations sur un large éventail de widgets (boutons, cartes, grilles, dialogues ...) au niveau des actions et gestes liés. En outre, des bonnes pratiques d'accessibilité sont également au programme. Enfin, au niveau typographie, la police Roboto introduite avec Ice Cream Sandwich est conservée, mais subit un petit lifting. La bonne nouvelle vient également du fait que Google propose de bénéficier du Material Design sur le Web via l'intégration dans le projet Polymer. Une fois la preview d'Android L installée, il est possible de bénéficier du Material Design en spécifiant que le thème de l'application doit hériter de Material Design dans le fichier styles.xml :

```
<resources>
    <style name="AppTheme" parent="android:Theme.Material">
    <!-- ... -->
    </style>
</resources>
```

À l'instar du thème Holo, le thème Material existe en version sombre (Dark Material), claire (Light Material) (Fig.2), mais également en version claire avec une barre d'action sombre (Dark ActionBar Material).

Lors de la construction d'interfaces s'appuyant sur Material Design, il est conseillé de privilégier au maximum l'affichage en grille par nature plus responsive et de garantir un espacement adéquat entre les éléments visuels afin d'avoir une zone de toucher optimale. D'autre part, la notion d'élévation apparaît sur les widgets et permet de définir l'ombre associée, mais également l'ordre d'affichage en profondeur.

Au niveau widgets, le Material Design amène 2 nouveautés : le CardView et le RecyclerView. Le premier nommé est une extension du FrameLayout offrant un affichage de contenu à l'intérieur de cartes identiques à celles de Google Now. Il s'utilise comme suit :

```
<android.support.v7.widget.CardView
  xmlns:card_view="http://schemas.android.com/apk/res-auto"
  android:id="@+id/card_view"
  android:layout_gravity="center"
  android:layout_width="200dp"
  android:layout_height="200dp"
  card_view:cardCornerRadius="4dp">
  <TextView
    android:id="@+id/info_text"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
</android.support.v7.widget.CardView>
```

L'objet RecyclerView est une version plus avancée et flexible de la classe ListView. C'est donc un conteneur permettant d'afficher un grand nombre de vues sous forme de liste avec une gestion du recyclage des vues et du scrolling automatique. Le RecyclerView est donc à utiliser en priorité dès lors que l'on affiche une liste d'éléments qui change dynamiquement. S'appuyant sur un LayoutManager pour le positionnement des items, il propose des animations par défaut pour les opérations classiques sur les items. La Fig.3 donne un bon aperçu de son architecture.

Pour fonctionner correctement, un RecyclerView nécessite un LayoutManager, servant à positionner les items et à réutiliser les vues des items non visibles, et un Adapter faisant le lien avec les données. Le LayoutManager

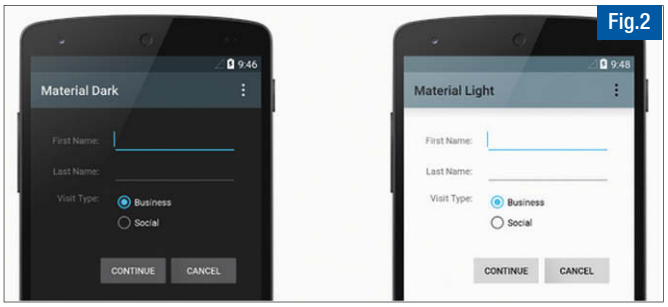


Fig.2

Versions Dark et Light

Fig.3

Architecture du RecyclerView

garantit des performances supérieures puisqu'il évite la création de vues superflues et les appels inutiles à findViewById. Concrètement, on l'utilise de la sorte :

```
// Layout
<android.support.v7.widget.RecyclerView
  android:id="@+id/my_recycler_view"
  android:scrollbars="vertical"
  android:layout_width="match_parent"
  android:layout_height="match_parent"/>

// Activité
public class MyActivity extends Activity {
  private RecyclerView mRecyclerView;
  private RecyclerView.Adapter mAdapter;
  private RecyclerView.LayoutManager mLayoutManager;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.my_layout);
    mRecyclerView = (RecyclerView) findViewById(R.id.my_recycler_view);
    mRecyclerView.setHasFixedSize(true);
    mLayoutManager = new LinearLayoutManager(this);
    mRecyclerView.setLayoutManager(mLayoutManager);
    mAdapter = new MyAdapter(myDataset);
    mRecyclerView.setAdapter(mAdapter);
  }

  ...
}

// Adapter simple
public class MyAdapter extends RecyclerView.Adapter<MyAdapter.
ViewHolder> {
  private String[] mDataset;

  public static class ViewHolder extends RecyclerView.ViewHolder {
    public TextView mTextView;
    public ViewHolder(TextView v) {
      super(v);
      mTextView = v;
    }
  }
```

La plus belle des rentrées avec PROGRAMMEZ! le magazine du développeur

Abonnez-vous !

Offre spéciale rentrée 2014 (1)

Programmez! et les éditions ENI vous plongent dans l'univers du Raspberry Pi pour 1 € de plus

(valeur du livre : 29,90 € - livre numérique - email indispensable)

(1) Valable uniquement pour la France métropolitaine. Quantité limitée. L'abonnement doit être envoyé avant le 27 septembre

Offre réservée à la France Métropolitaine

Spécial étudiant
39 €
1 an 11 numéros

1 an 11 numéros
49 €
seulement (*)

2 ans 22 numéros
79 €
seulement (*)

(*) Tarifs France métropolitaine

© SensorSpot / istock

© fotografiedk - Fotolia.com

Toutes nos offres sur www.programmez.com

Oui, je m'abonne

ABONNEMENT à retourner avec votre règlement à Programmez, 17, route des Boulangers 78926 Yvelines cedex 9

☐ ***Abonnement 1 an au magazine + livre numérique ENI "Raspberry Pi" :*** 50 € au lieu de 65,45 € / prix au n° + 29,90 €) : 50 € / abonnement seul : 49 €

☐ ***Abonnement 2 ans au magazine + livre numérique ENI "Raspberry Pi" :*** 80 € au lieu de 130,90 € / prix au n° + 29,90 €) : 80 € / abonnement seul : 79 €

☐ ***Abonnement spécial étudiant 1 an au magazine + livre numérique ENI "Raspberry Pi" :*** 40 € au lieu de 65,45 € / prix au n° + 29,90 €) : 40 € / abonnement seul : 39 €
Photocopie de la carte d'étudiant à joindre

☐ M. ☐ Mme ☐ Mlle Entreprise : Fonction :

Prénom : Nom :

Adresse :

Code postal : Ville :

Tél : ***(Attention, e-mail indispensable pour les archives sur internet)***

E-mail : @

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

PROG 177
Offre limitée, valable jusqu'au 27 septembre 2014

```
public MyAdapter(String[] myDataset) {
  mDataset = myDataset;
}

@Override
public MyAdapter.ViewHolder onCreateViewHolder(ViewGroup parent,
                           int viewType) {
  View v = LayoutInflater.from(parent.getContext())
              .inflate(R.layout.my_text_view, parent);
  // ...
  ViewHolder vh = new ViewHolder(v);
  return vh;
}

@Override
public void onBindViewHolder(ViewHolder holder, int position) {
  holder.mTextView.setText(mDataset[position]);
}

@Override
public int getItemCount() {
  return mDataset.length;
}
}
```

Android Wear

Les objets connectés sont à l'honneur avec Android Wear. Déclinaison d'Android dédiée aux wearables, cette version fait son apparition dans des montres intelligentes également appelées smartwatches. Celles-ci permettent l'exécution d'application tout en donnant accès à des fonctionnalités hardwares telles que l'accéléromètre. Leur principal avantage étant de pouvoir être associées à des smartphones partageant les notifications et intégrant Google Now notamment. Google, Samsung et LG sont les premiers acteurs à commercialiser ce type d'appareils sous Android Wear. Fondamentalement, une application Android Wear est une application Android classique prenant en compte les spécificités induites par les wearables en termes d'IHM, mais également de temps d'utilisation. Ainsi, seules 5 API du SDK Android ne sont pas supportées :

- android.webkit
- android.print
- android.app.backup
- android.appwidget
- android.hardware.usb

Pour développer une application Android Wear, il est préférable de passer par Android Studio qui propose des wizards facilitant la création. Un émulateur Android Wear est également mis à disposition. À charge désormais aux développeurs de mettre à profit ce nouveau SDK pour créer des applications exploitant les nouveaux cas d'usage liés aux wearables.

Android TV

Apparu en 2010, le système Google TV n'aura jamais su trouver son public du fait des limitations de la plateforme empêchant les développeurs de proposer aux utilisateurs des applications dignes d'intérêt. Fort de ce constat, Google tente une nouvelle incursion dans le secteur avec Android TV qui peut être installé au sein de télévisions ou de set-top boxes. Android TV est une plateforme permettant d'accéder au Google Play Store et donc au téléchargement d'applications et de jeux. Les services de streaming seront également proposés ce qui permettra aux utilisateurs de profiter de Netflix ou Hulu. La recherche vocale est au programme d'une plateforme promise à un bien meilleur avenir que sa devancière.

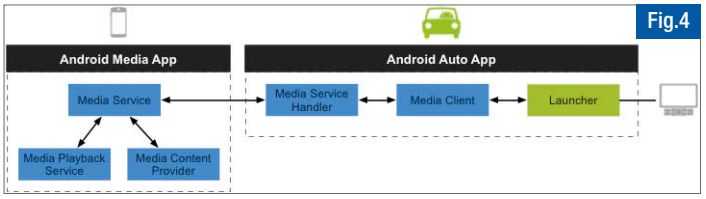


Fig.4

Architecture Android Auto

Enfin, la technologie Google Cast, utilisée par le Chromecast, est également supportée. Les applications pour télévisions utilisent le même modèle que celles pour smartphones et tablettes. Il est ainsi possible de modifier des applications existantes pour les faire tourner sous Android TV. Pour ce faire, il y a 2 étapes essentielles. La première implique de déclarer une activité dans le manifest qui servira de lanceur de l'application dans la télévision en utilisant un filter sur la catégorie android.intent.category.LEANBACK_LAUNCHER.

Il conviendra d'apporter une attention particulière aux layouts des applications ciblant les télévisions, du fait des spécificités de leurs écrans, en suivant les guidelines Google sur le sujet. La seconde étape consiste ensuite à ajouter les bibliothèques de support Android TV qui fournissent les API et widgets spécifiques et qui se trouvent dans la preview proposée par Google. Nommées leanback v17 et recyclerview v7, ces bibliothèques ne sont pas obligatoires, mais fortement recommandées afin de réaliser des applications optimisées pour les télévisions. Pour exécuter une application Android TV, le développeur pourra utiliser soit l'émulateur habituel en définissant un périphérique de type TV soit une véritable télévision.

Android Auto

Dernier moment fort de Google I/O, l'annonce de la sortie d'Android Auto, la version d'Android pour les automobiles. Pour soutenir Android Auto, Google a lancé un consortium en début d'année 2014 qui regroupe déjà une quarantaine de constructeurs parmi lesquels Hyundai, Audi ou Volvo. Les premières voitures équipées sont attendues pour début 2015. Techniquement parlant, Android Auto offre une expérience d'Android optimisée au sein d'un véhicule lorsqu'un utilisateur y connecte un périphérique compatible. Toute l'intelligence de la plateforme se situe donc au niveau de ce dernier et l'affichage de la voiture devient une interface projetée du périphérique. L'utilisateur interagit ensuite avec le système via des contrôles vocaux ce qui place Google Now au cœur du système.

Android Auto fournit un modèle UI simple se concentrant autour de 4 axes : interface utilisateur multimédia supportant des applications vidéo ou musicales, notifications, actions vocales et workflow de développement simplifié. Enfin, un mot sur l'architecture d'Android Auto (Fig.4). Elle se compose de 3 composants principaux : l'Android Media App qui fournit le contenu à l'Android Auto App, l'Android Auto App qui crée l'UI à destination de l'écran du véhicule et gère les interactions avec l'utilisateur, et enfin l'écran au sein du véhicule. Pour fonctionner, l'Android Media App doit implémenter des binders pour les APIs Browsing et Playback.

Développer des applications pour Android TV implique de télécharger la preview du SDK alors que l'exécution en phase de test nécessitera l'emploi d'un périphérique simulant l'écran d'un véhicule. Reste ensuite à installer l'APK Android Auto sur ce périphérique et à installer l'APK de l'application sur un smartphone.

Conclusion

L'avenir pour les développeurs Android s'annonce radieux à condition de mettre à profit les nouvelles opportunités proposées par Google pour réaliser des applications innovantes répondant à des cas d'usage inédits qui apparaîtront avec ces nouveaux types de supports.

Sylvain **Saurel** – *Ingénieur d'Etudes Java / JEE*
sylvain.saurel@gmail.com

BEST-SELLER

Aspose.Words for .NET | à partir de € 718

ASPOSE Your File Format APIs

Lisez, modifiez et écrivez des documents Word sans Microsoft Word.

- Création de documents, manipulation du contenu/formatage, puissante capacité de fusion de courrier et exportation en DOC/HTML
- Accès détaillé à tous les éléments d'un document par programmation
- Support les formats de fichiers: DOC, DOCX, WordprocessingML, RTF, HTML, OOXML, OpenDocument, PDF, XPS, EMF et EPUB

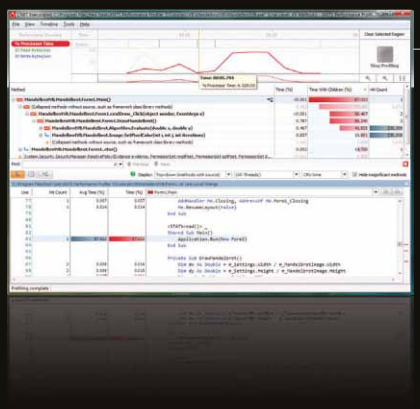
BEST-SELLER

oXygen XML Editor Professional | à partir de € 350

<oXygen/> XML Editor

Éditeur XML multiplateforme supportant la plupart des technologies XML.

- Distribuez/actualisez facilement des plug-in/frameworks pour Oxygen
- Simplifiez la configuration des projets XML avec Master Files
- Affichage des modifications et commentaires de révision dans légendes
- Support d'édition visuelle convivial pour DocBook, DITA, TEI, XHTML
- Validez les documents XML avec les schémas XML, Relax NG, DTD, NVDL et Schematron



Red Gate .NET Developer Bundle | à partir de € 760

redgate®

Éradiquez et corrigez le code lent, identifiez le code .NET bogué et comprenez les raisons.

- Bénéficiez d'une vue d'ensemble des performances de vos applications et identifiez les goulots d'étranglement, dans le code ou la base de données
- Trouvez rapidement les fuites de mémoire et optimisez la mémoire de vos codes C# et VB.NET
- Comprenez et déboguez le code de tiers, frameworks, composants et bibliothèques inclus
- Standardisez la gestion des performances de votre équipe de développement

BEST-SELLER

DevExpress DXperience | à partir de € 1 077

DevExpress

Tous les outils DevExpress ASP.NET, WinForms, Silverlight, WPF et IDE Productivity en un.

- Abonnement de 12 mois pour tous les produits et mises à jour DevExpress et accès aux versions bêta en développement actif
- Modèles et thèmes d'application intégrés exceptionnels
- Support de nouvelle vue Windows 8 UI et panneaux ancrables tactiles
- Support interface codée pour test environnement utilisateur

Siège social en Europe
ComponentSource
30 Greyfriars Road
Reading
Berkshire
RG1 1PE
Royaume-Uni

Siège social aux États-Unis
ComponentSource
650 Claremore Prof Way
Suite 100
Woodstock
GA 30188-5188
États-Unis

Siège social au Japon
ComponentSource
3F Kojimachi Square Bldg
3-3 Kojimachi Chiyoda-ku
Tokyo
Japon
102-0083

Numero vert:
0800 90 92 62
www.componentsource.com

Nous acceptons les bons de commande. Contactez-nous pour demander un compte de crédit.

VISA MasterCard

septembre

PASS SQL Saturday

Une journée dédiée à SQL Server se déroulera le 13 septembre à Paris. Organisée par la communauté française SQL Server, cette journée promet d'être riche : + 20 sessions techniques. Et les sujets seront très nombreux : haute disponibilité, cloud, mobilité, migration, BI et Big Data, etc. Site : http://guss.pro/2014/09/13/agenda-du-sqlsaturday-paris-2014/

Droidcon Paris 2014

Les 22 et 23 septembre prochains, la grande conférence Android se tiendra à Paris : la Droidcon. L'appel aux conférences s'est clos le 15 juillet dernier. La conférence est coorganisée par le PAUG et BeMyApp.

octobre

2 octobre : ReBuild à Nantes

Le 2 octobre, un grand événement communautaire autour des technologies Microsoft se tiendra à Nantes : 40 sessions techniques, 38 intervenants, 15 partenaires. C'est l'occasion de voir et de revoir des technologies et des outils Microsoft en dehors des événements parisiens. Cette année, on pourra voir Azure, SharePoint, Visual Studio Online, design d'interface, Kinect, SQL Server, Xamarin, Office 365, etc.
Site : https://lescommunautesms-public.sharepoint.com/rebuild

7 octobre : IBM SolutionsConnect 2014 à Paris

IBM organisera une journée de conférences le 7 octobre à la Cité des Sciences et de l'Industrie. De nombreux thèmes seront abordés : big data, cloud, mobile, sécurité... Une partie développeur sera organisée : DeveloperConnect avec 3 grands thèmes (devops, ingénierie en continu et mobilité).
Site : http://goo.gl/Co7dHk

10 octobre : dotGo

Vous connaissez peut-être le langage Go. Une grande conférence européenne se déroulera à Paris le 10 octobre prochain. Au menu : conférences, ateliers. Une excellente occasion pour découvrir ce langage et d'échanger avec les développeurs l'utilisant.
Site : http://www.dotgo.eu

23 & 24 octobre : Forum PHP Paris 2014

Le grand événement de communauté PHP revient fin octobre à Paris. Cinq thèmes seront montrés : agilité, devops, écosystème PHP, cloud et retour d'expérience. Des journées à ne pas manquer.
Site : http://afup.org/pages/forumphp2014/

29 & 30 octobre : Blend Web Mix à Lyon

Durant deux jours, Lyon accueille une conférence Web : conférences, ateliers, rencontres avec les startups, speed démo, soirées, etc. C'est l'occasion de voir quelques projets de recherches autour du Web et du futur du Web. L'édition 2013 avait rassemblé +850 personnes ! site : http://www.blendwebmix.com

Prochainement...

Green Code Lab Challenge 2014 : l'événement de l'éco-informatique aura lieu du 26 au 28 novembre. Comment faire un code et un logiciel écoresponsable ? Toutes les réponses au Green Code Lab ! site : http://www.greencodelab-challenge.org/GCL2014/

Drupagora 2014 : l'événement français Drupal se déroulera le 14 novembre à Paris. Ce sera déjà la 4e édition. Site : http://www.drupagora.com/

dotJS + dotCSS : le 15 novembre, une double journée technique sur JavaScript et les CSS. Les ateliers seront les vedettes de cette journée. Site : http://www.dotjs.eu/workshops

JDev 2015 : pas de conférence JDev (journées nationales du développement logiciel de l'enseignement supérieur et recherche) cette année. Ces journées se tiendront du 30 juin au 3 juillet 2015 à l'Institut Polytechnique de Bordeaux. Site : http://devlog.cnrs.fr/jdev2015

Programmez ! arrive sur Windows Store

Programmez ! est désormais disponible sur Windows Store. Notre application gratuite permet d'accéder aux derniers PDF de la revue. Fonctionne sur Windows 8.x. D'autres applications seront disponibles très prochainement. L'application Programmez ! a été développée par InfiniteSquare.

Montée en charge linéaire et extrêmement performante

NCache™
100% .NET

Pour applications .NET et Java
(supporté sur Windows Azure et Amazon AWS)

TayzGrid™
100% Java

Les données en cache (via NCache), réduisent les accès coûteux en base, et permettent à vos applications de monter en puissance vers “extreme transaction processing” (XTP). TayzGrid est une implémentation native 100% Java de NCache.

Cache distribué en mémoire

- Extrêmement rapide et montée en charge linéaire avec 100% uptime
- Topologie en Miroir, Répliquée, Partitionnée et Cache Client
- NHibernate et Entity Framework cache niveau 2

Optimisation ASP.NET de Web Farms

- ASP.NET Session State cache
- ASP.NET View State cache
- ASP.NET Output Cache provider

Memcached Wrapper Fourni

Partage de données en mode Runtime

- Notifications d'événements puissants pour le partage de pub / sub données

Alachisoft
sales@alachisoft.com
US: +1 (925) 236 3830
Siège de l'entreprise

Télécharger un essai GRATUIT!
www.alachisoft.com

RedFabriQ
info@redfabriq.com
Tel: +33 1 40 16 07 89
Distributeur et intégrateur en France

Écoconception web

2e partie

Conception : les atouts de la frugalité

Les étapes qui précèdent la phase de développement sont les plus importantes pour réduire les impacts environnementaux d'un logiciel ou d'un site web. Il faut notamment réduire la couverture fonctionnelle à l'essentiel pour faire fondre le gras numérique.

Un nombre croissant de logiciels et de sites web sont tellement « obèses » qu'ils posent des problèmes de performance et / ou de coût aux entreprises qui les exploitent. Avec leur montée en charge, certaines entreprises et administrations sont confrontées à de véritables défis techniques pour continuer à les faire fonctionner. Au point qu'un nouveau mot est apparu dans la langue française pour désigner cette tendance : obésiciel. Né de la contraction entre « obèse » et « logiciel », il permet d'identifier ces applications qui nécessitent des quantités démesurées de ressources informatiques – mémoire vive, cycles processeur, espace de stockage, bande passante, etc. – pour fonctionner.

Heureusement, ce phénomène n'est pas inéluctable et, comme dans la vraie vie, il est possible de faire faire un régime aux obésiciels pour qu'ils perdent leur « gras numérique ».

Cependant, l'idéal consiste surtout à adopter un bon régime alimentaire dès le début, c'est-à-dire de pratiquer une démarche d'écoconception logicielle. Cette approche d'amélioration continue vise à appliquer des principes d'efficience, de simplicité et de sobriété tout au long du cycle de vie du logiciel.

Evidemment, les étapes de spécification du besoin et de conception sont essentielles. Ce sont en effet elles qui concentrent les plus gros effets de levier. Car, optimiser le code d'une fonctionnalité qui ne sera jamais utilisée n'est certainement pas l'approche la plus constructive...

Dans cette nouvelle série d'articles, nous allons vous présenter un ensemble de bonnes pratiques d'écoconception logicielle appliquée aux sites web et aux services en ligne. Ces bonnes pratiques sont en grande partie tirées du livre « éco-conception web : les 100 bonnes pratiques » paru chez Eyrolles et dont nous sommes les auteurs.

Pour commencer, nous allons aborder dans cet article toutes les étapes qui précèdent la création du code : spécification, conception fonctionnelle et graphique, conception technique, etc. L'objectif est de vous aider à mettre en place des réflexes qui vous permettront de créer des applications plus sobres, plus légères, et plus simples à utiliser. Et donc, au final, dont les impacts environnementaux seront réduits au minimum.

Viser une couverture fonctionnelle minimaliste

Nous aurions tendance à dire, si le rédacteur en chef nous le permettait, que ce n'est pas celui qui a la plus grosse (voiture) qui fait pipi le plus loin ! Dit autrement, l'utilisation d'un marteau pour écraser un moustique, semble un tantinet exagérée. C'est pourtant une pratique courante en 2014. Les directions métier régnant en maîtres absolus sur le système d'information et le client ayant toujours raison, les applications métier ont tendance à faire du gras fonctionnel. Le problème, c'est que ce gras se traduit par une complexification dangereuse des projets, un surcoût d'exploitation et l'aggravation inutile de la dette technique du logiciel (coût de maintenance évolutive et corrective notamment). La couverture fonctionnelle doit donc être centrée sur le besoin essentiel des utilisateurs. Prenons deux exemples pour illustrer l'impact d'une (trop) large couverture fonctionnelle. Office 2010 nécessite 512 Mo pour fonctionner(*) alors qu'ABIWorld consomme moins de 64 Mo et Wordpad moins de 10 Mo. Ces deux derniers outils sont suffisants dans plus de 80 % des usages car ils intègrent les fonctionnalités essentielles : mise en forme, pagination, insertion d'images et de tableaux, table des matières, notes de bas de page, etc. Dans un autre registre, le lecteur PDF Adobe Reader nécessite au minimum 256 Mo de mémoire vive pour fonctionner(*) tandis que Foxit Reader se contentera de quelques dizaines tout au plus. Le premier met tellement de temps à démarrer que certains utilisateurs installent le second gratuit, même s'ils ont payé la licence du premier, pour arrêter de perdre du temps. Encore une fois, Foxit Reader est largement suffisant pour lire des PDF, ce qui représente 90 % des usages. Et, si besoin, il est possible d'ajouter des extensions payantes pour obtenir la même couverture fonctionnelle que les obésiciels d'Adobe.

Le constat est exactement le même pour un site web ou un service en ligne, tant côté serveur que dans le navigateur de l'internaute. A la fin des années 1990, Google s'est imposé face à Yahoo! par sa couverture fonctionnelle minimaliste qui se résumait à un champ ! Cette sobriété lui a permis, à l'époque, d'afficher des résultats bien plus rapidement que ses principaux concurrents - Yahoo !, Hotbot, etc.

(*) configuration minimale requise indiquée par Microsoft sur son site

Fig.1

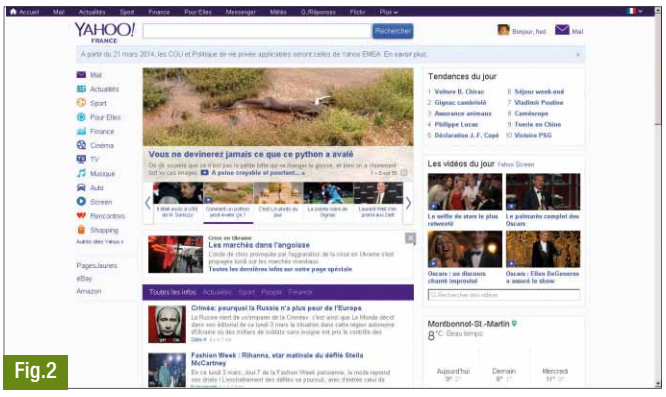


Fig.2

Illustration : google / yahoo : la simplicité de Google a fait son succès face à Yahoo !

Septembre

Casual Games Cup : testez les jeux !

Du 1er au 15 septembre, les joueurs du site jeux.com pourront tester les jeux réalisés et développés durant la Casual Games Cup. Il s'agit d'un concours national pour créer des « casual games ». Chaque projet fournit le jeu, un descriptif. La remise des prix aura lieu début octobre à Bordeaux. Site : http://www.casualgamescup.com/

Octobre

Code of War 3e édition

Cette nouvelle édition se déroulera à Bordeaux le 17 octobre. « Code of War est une compétition de programmation en équipe, gratuite et ouverte à tous. Deux ou trois personnes peuvent composer une équipe, dont au moins une personne avec des notions de programmation. Ce tournoi ne va pas récompenser le meilleur programmeur, mais la meilleure tactique. Chaque équipe aura à programmer l'IA de leur robot au travers du kit de développement disponible pour l'occasion. Stratégie et tactique de jeux sont les compétences nécessaires pour s'assurer la victoire. » précise les organisateurs. Les développeurs individuels peuvent participer à la compétition. La plateforme de jeux est d'ores et déjà disponible. Foncez ! site : http://www.codeofwar.net/

Scala.IO : la conférence Scala !

ScalaIO est une conférence organisée par des passionnés de la communauté Scala. Pour sa seconde édition, ScalaIO se deroulera les 23 et 24 octobre prochains et accueillera à Paris plus de 300 personnes venues de toute l'Europe pour assister aux présentations de près de 40 speakers talentueux, répartis comme l'an dernier sur 3 tracks. Au moins 30% des présentations seront données en Anglais et nous assurons une track pour un public anglophone. L'événement aura lieu dans les Salons du Tapis Rouge, un espace prestigieux situé au coeur de Paris en face de la mairie du Xème. Le lieu disposera d'une grande salle de conférence en sous-sol pour accueillir les keynotes et la track anglaise, d'une salle de 200 places et d'une salle de 80 places pour les 2 autres tracks.
ScalaIO est une conférence centrée principalement sur le langage Scala mais le big data processing, les retours d'expériences, les ateliers, les architectures distribuées seront à l'honneur. Début septembre, l'agenda complet sera disponible. Site : http://scala.io/

Novembre

Mobile Dev Day à Mons (Belgique)

Le 27 novembre, venez découvrir le développement mobile et embarqué sur les plateformes Microsoft. Les principaux thèmes seront : mobilité, interface et design, cloud computing et internet des objets. 11 sessions seront jouées durant la journée. Pour en savoir plus : www.mobiledevday.be

– qui proposaient tous un grand nombre de services associés à la recherche : réservation d'avion, météo, e-mail, etc. Sauf que... les internautes cherchaient uniquement des sites web... Depuis, les grands succès du web moderne, de Twitter à Instagram en passant par WhatsApp reposent sur cette sobriété volontaire. 140 caractères pour Twitter... on peut difficilement faire plus simple et sobre ! Fig.1 et 2.

Dans un autre registre, et toujours dans le domaine du web, MySQL ne doit pas son succès qu'à sa gratuité. La base de données s'est imposée au début des années 2000 par sa rapidité grâce à son moteur MyISAM qui ne supportait pas les transactions ACID. Quel intérêt pour afficher des pages HTML ?

Bref, vous l'aurez compris, la sobriété fonctionnelle est l'un des piliers de l'écoconception web.

Cette approche minimaliste permet même de faire travailler les utilisateurs à votre place ! Par exemple, tous les trucs inventés par les utilisateurs pour enrichir le service – les # et @ de Twitter par exemple - reposent sur un consensus entre eux. Difficile de faire autant preuve du sens de l'écoute de ses clients / utilisateurs Fig.3.

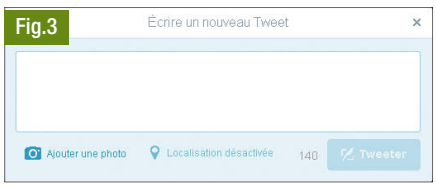
Quantifier et typer précisément le fonctionnel

La sobriété fonctionnelle doit aussi s'accompagner d'une description et d'une quantification très précises des données manipulées par l'application. Un degré de précision d'autant plus facile à atteindre que la couverture fonctionnelle est restreinte. Inutile en effet de stocker en BIG INT des données qui peuvent l'être en SMALL INT. Encore faut-il que le développeur et / ou l'architecte logiciel disposent des informations suffisantes pour prendre cette décision si elle n'est pas précisée. Cette logique suit les données tout au long de leur cycle de vie, notamment ensuite lors de leur manipulation au sein du serveur d'application, puis du transfert et de la manipulation dans l'interface utilisateur (en Javascript donc).

Le raisonnement se poursuit avec la quantité de données. Inutile de retourner et d'afficher 25 enregistrements lorsque l'utilisateur n'exploitera que les 10 premiers. Une petite clause LIMIT fera le plus grand bien à votre site web. C'est vrai pour un moteur de recherche, un site de commerce électronique, etc. Microsoft Research a ainsi démontré qu'en évitant d'afficher 20 % de résultats sur son moteur de recherche Bing, il économise 80 % de la consommation électrique côté serveur. Autrement dit, on observe des effets de levier importants et non linéaires en fonction du volume de données manipulées Fig.4.

Plus généralement, le niveau de qualité de service attendue doit être précisé. C'est autant valable pour la qualité (et donc le poids) des images publiées par un CMS que pour un temps de réponse, la richesse graphique d'un e-mail (format HTML ou texte ?), ou bien encore la précision d'un calcul. Inutile d'activer la virgule flottante lorsque l'on compte des clients ou des produits...

Fig.3



La sobriété fonctionnelle de Twitter a fait son succès.

Une interface utilisateur sobre

La sobriété fonctionnelle doit se traduire, en termes d'ergonomie, par une interface graphique utilisateur (GUI) dépouillée. Il faut notamment s'appuyer sur les possibilités de HTML5 et CSS3 pour imaginer le design du site plutôt que d'imaginer un site dans l'absolu et d'essayer ensuite de le réaliser. Cette frugalité met en valeur les fonctionnalités essentielles de l'application qui en devient ergonomique par sa simplicité. Lorsqu'un site web ne vous propose qu'un seul champ de saisie, sans fioriture graphique, il est difficile d'hésiter !

Toujours dans cette logique de simplicité, il faut essayer d'utiliser ce qui existe déjà. Par exemple, les polices standards sont déjà présentes sur le poste client qui n'a donc pas besoin de les télécharger. Les navigateurs récents offrent également des services avancés sur lesquels il faut s'appuyer en priorité plutôt que d'utiliser des plug-in tels que Flash, applet Java, Silverlight, etc. A part pour du contenu multimédia, il est franchement rare que HTML 5 et ECMA Script ne suffisent pas.

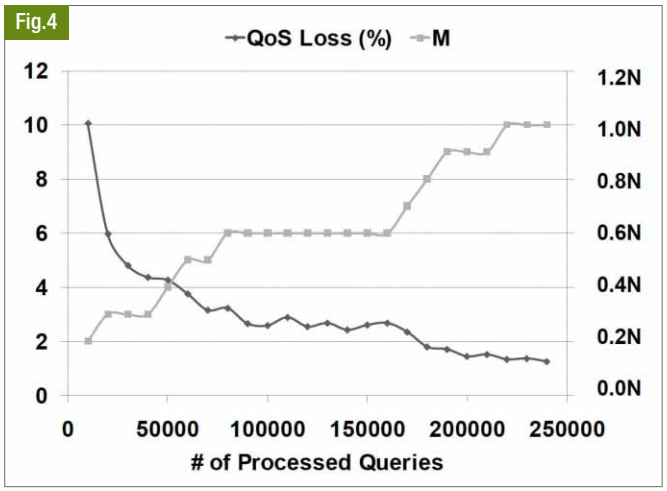
Enfin, les internautes s'appuient sur une sorte de code de la route implicite pour naviguer sur les pages web. Il faut donc respecter des principes standards tels que la navigation rapide dans l'historique. Cette fonction évite de redemander la page au serveur puis de la recharger (gain en requêtes HTTP et bande passante). Evitez par conséquent les manipulations rendant la page non utilisable après l'avoir quittée Fig.5.

Ces bonnes pratiques peuvent paraître simplistes. Mais combien les appliquent réellement au quotidien et dans leur ensemble ?

Frédéric **Bordage**, *expert écoconception logicielle, @greenit*

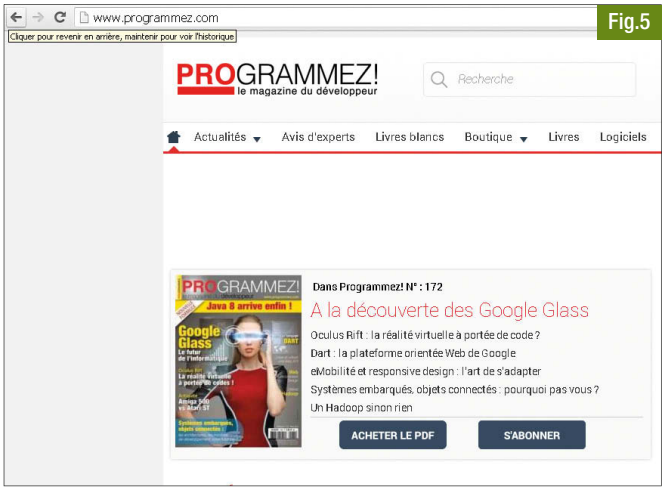
Jérémy **Chatard**, *directeur technique de Breek.fr, @breek*

Fig.4



Microsoft Research indique qu'en réduisant de 20 % le nombre de résultats affichés par son moteur de recherche, la consommation électrique côté serveur est réduite de 80 %

Fig.5



Le recours aux standards de navigation, notamment les fonctionnalités natives du navigateur web, réduisent l'empreinte ressources tout en améliorant le niveau d'accessibilité du site.

ikoula
HÉBERGEUR CLOUD

LE CLOUD GAULOIS, UNE RÉALITÉ ! VENEZ TESTER SA PUISSANCE

EXPRESS HOSTING

Cloud Public
Serveur Virtuel
Serveur Dédié
Nom de domaine
Hébergement Web

sales@ikoula.com
01 84 01 02 66
express.ikoula.com

ENTERPRISE SERVICES

Cloud Privé
Infogérance
PRA/PCA
Haute disponibilité
Datacenter

sales-ies@ikoula.com
01 78 76 35 58
ies.ikoula.com

EX10

Cloud Hybride
Exchange
Lync
Sharepoint
Plateforme Collaborative

sales@ex10.biz
01 84 01 02 53
www.ex10.biz

Découverte et apprentissage de la programmation à l'école

© iStock

Depuis quelques mois, il y a une frénésie autour de la programmation, du métier de développeur. Les déclarations gouvernementales se succèdent, comme si, on venait de découvrir le sujet et les problèmes (on manque de développeurs et de professionnels du numérique). Partout, on veut que le développeur soit reconnu.
Deux mouvements se profilent :

1 la découverte ludique de la technologie aux enfants par des ateliers, des conférences comme nous avons pu en avoir au Devoxx4Kids, à la Maker Faire Paris, à l'atelier Nao, etc.
2 la découverture et l'apprentissage de la programmation à l'école pour les enfants et les adolescents. Ces actions sont plus ou moins ponctuelles et souvent portées par des associations comme Fier d'être dév.

Le développeur est à la base de l'économie numérique actuelle et de demain. Sans codes pas d'applications. Sans développeurs, pas de codes ! CQFD. Plusieurs personnalités poussent à apprendre la programmation même si ce n'est pas pour en faire un métier, mais pour savoir comment fonctionnent les choses. Barack Oboma soutient les initiatives américaines tout comme Richard Branson. Code for America est un des exemples les plus importants aux États-Unis. En France, l'initiative « Code for France » a été lancée, mais pour le moment, ce projet a du mal à être visible.
Code.org propose une approche originale : en 1 heure, il propose de découvrir les bases du code avec les oiseaux et les cochons. C'est ludique, très drôle et instructif. On utilise des blocs pour construire la logique et l'outil génère derrière le code.
La programmation aide à acquérir une certaine logique, une structuration de la pensée et de l'action. Mais après il y a apprendre et apprendre. Il ne faut pas imposer une découverte technique de la programmation, c'est le meilleur moyen de rebuter un enfant et un adolescent. L'objectif doit être à la fois ludique et pédagogique : on fait découvrir avec des notions simples le code et ce que le code peut faire à un robot ou comment on construit un jeu, un programme.
Cette notion de découverte est importante, car après c'est le choix de l'enfant, de l'ado de continuer ou non dans cette voie. Certains le feront, d'autres non. Mais cela doit rester son choix. Ne nous leurrons pas, une majorité d'entre eux, seront des consommateurs du numérique et rien de plus.
Cette envie du numérique et de codes doit venir de l'enfant.
Après, si un enfant de 10-12 ans veut continuer dans la programmation, avec un minimum de matériels et d'outils, il peut apprendre par lui-même, les ressources en ligne, les livres (et Programmez !) sont nombreux et facilement accessibles. Des tarifs éducations peuvent réduire la facture.
À Programmez !, nous sommes attentifs à ces initiatives. Et nous ne pouvons que les encourager.

François **Tonic**

La tentation du code obligatoire dès le primaire

Depuis quelques mois, les déclarations se succèdent : introduction de la programmation à l'école, loi sur l'obligation d'apprendre le code dès le primaire, former des bataillons de développeurs, etc.

Au préalable, nous pouvons dresser plusieurs constats :

- Un manque de développeurs en France : attention tout de même à ne pas généraliser à tous les profils,
- La France doit renforcer et développer son économie numérique,
- Des secteurs peu soutenus ou peu visibles : nous possédons de fortes compétences en robotique et en objets connectés,
- Les prochaines générations doivent être acteurs du numérique et pas des consommateurs lambda : bref, savoir comment fonctionne la technologie pour mieux l'utiliser.

Un récent sondage (Syntec Numérique – BVA) dit que les Français sont favorables à 87 % pour l'apprentissage de la programmation à l'école (à partir du collège essentiellement). Ce chiffre vaut ce qu'il vaut. Nous sommes assez circonspects sur sa pertinence et la perception des sondés sur la notion de programmation, de code...

« Apprendre le code et la programmation aux enfants »

Une fois énoncé, nous avons tout dit et rien dit. Car entre les déclarations « nous devons », « il faut » et la réalité, le fossé est toujours aussi abyssal. En France, les filières scientifiques souffrent énormément avec une désaffection des élèves. Les filles sont peu attirées (même si la situation s'améliore). L'informatique en fait partie.

L'économie actuelle et future repose en partie sur le numérique, donc de l'informatique, donc des programmes. Et il faut des développeurs, des codeurs. Or, nous manquons clairement de compétences numériques et de développeurs. Autre constat, mieux sensibiliser les enfants au numérique, à l'informatique, pas seulement pour qu'ils deviennent informaticiens, mais pour qu'ils maîtrisent le numérique, comprennent comment cela fonctionne. Bref, être un acteur actif de ce monde et non plus un consommateur passif. C'est un élément important à garder en tête. La programmation peut aider l'enfant à mieux comprendre le numérique, comment cela fonctionne, tout en apportant une forme de logique et de structure de la pensée. Car la programmation est un univers logique et structuré. Voir l'apprentissage du code à l'école avec pour seule finalité de former des programmeurs serait une erreur et pourrait produire l'inverse de l'effet voulu.

Oui, la découverte de la programmation peut éveiller les enfants à notre univers, voire, susciter des vocations futures. Mais cela n'aurait aucun sens de dire « demain, tous développeurs ». Car cet argument est, pour nous, absurde. Seule une minorité d'enfants s'orientera dans le code, même si dans le monde du travail, ils seront amenés à travailler de plus en plus avec, et pour, le numérique.

Avons-nous besoin de bataillons de programmeurs (terme que nous avons souvent lu) ? Non. Nous avons besoin de développeurs bien formés, avec de fortes compétences, et totalement polyglottes (sur les langages).

Où, quand, comment ?

Aujourd'hui, de nombreuses initiatives existent en France pour organiser, le plus souvent en dehors des heures de cours, des ateliers, des activités autour de l'initiation à la programmation. La réforme des rythmes scolaires permet en effet de mettre en place ces activités numériques. Des associations, parfois en soutien avec des éditeurs et constructeurs d'informatique, montent des « cours » et ateliers pour découvrir de manière ludique le code et la programmation, parfois, avec de la robotique. Des événements peuvent aussi avoir lieu, indépendamment du cadre scolaire / parascolaire, tels que Devoxx4Kid, l'atelier Aldebaran, Kids Coding, etc. Durant la Maker Faire Paris, des ateliers étaient dédiés au hack et aux codes.

Questions : l'apprentissage doit-il être obligatoire ? Doit-il se faire dans le cursus normal de l'école ? À partir de quel âge ? Qui fera cet apprentissage ? Et sur quels outils et matériels ?

Il y a débat pour savoir s'il faut inclure l'apprentissage du code (si celui-ci devient obligatoire) dans les heures de cours.

Dans ce cas, cela nécessitera un nouvel aménagement des cours et donc, sans doute, de réduire d'autres matières enseignées (sujet ô combien sensible). Ensuite, qui fera cet apprentissage ? Si ce sont les professeurs, il faudra les former, les sensibiliser, et cela lancera de nouveaux débats dans l'éducation nationale. Si ce sont des personnes extérieures : qui seront-elles ? Qui va les « certifier » ?

Se pose aussi un élément crucial : que veut-on faire apprendre et découvrir ? Est-ce le code purement et simplement ? Sa logique, sa structure, son fonctionnement ? Cet apprentissage, surtout si on parle à des enfants de 8-10 ans, doit être neutre et simplifié. Pas question de parler de classes objets, de Java, de C++, etc. Il faut travailler avec des langages et outils de type Scratch, Lego Mindstorm. Il est nécessaire d'avoir une approche ludique, pédagogique. Il faut que l'enfant voie en quelques clics, en quelques actions, qu'il peut manipuler un robot, un objet sur un écran, etc. Il faut faire abstraction de tout le reste. Ce n'est qu'après que l'enfant, l'adolescent pourront aller plus loin et commencer à rentrer dans le vif du code. Si cet apprentissage est trop technique ou mal adapté, l'enfant pourra vite être rebuté par l'informatique... Tout l'inverse de l'idée initiale. Cette découverte doit être la plus neutre possible sur les langages, les outils, le matériel. Plusieurs défis attendent l'école : la formation des professeurs, l'équipement des établissements scolaires (logiciels et matériels). Cela nécessitera un financement adapté. Et nous nous posons une autre question : qui définira le contenu de cet apprentissage ?

En 2013, Barack Obama s'adressait directement aux jeunes pour les inciter à s'impliquer dans le monde numérique, à découvrir la programmation. Il mettait en avant le volontarisme et les associations pour créer des formations, des ateliers (voir : https://www.youtube.com/watch?v=6XvmhE1J9PY). Peut-on imaginer une telle intervention en France ? Nous n'y croyons pas...

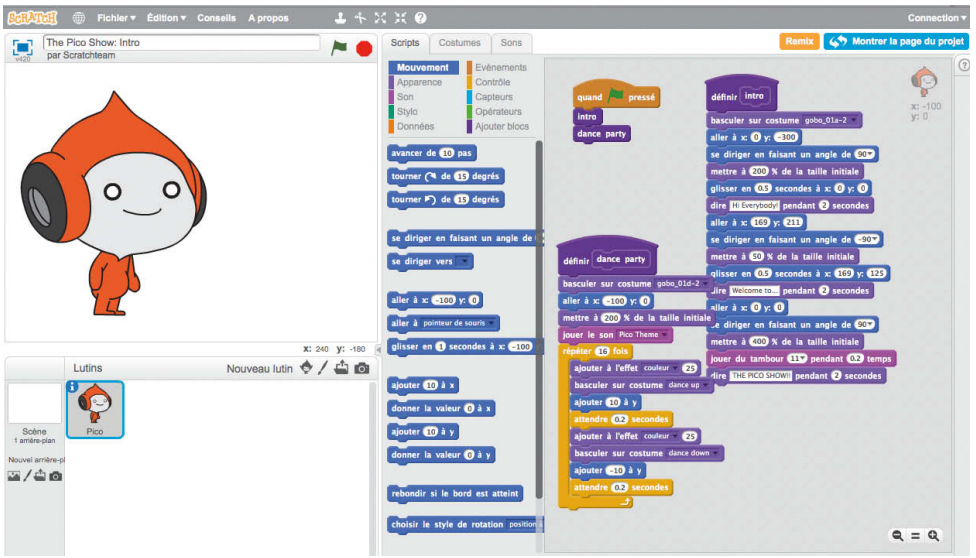
Quelques lectures

Slides conférence : http://goo.gl/yrSnA7
Codecademy : http://www.codecademy.com/fr
Article dans Slate.fr : http://goo.gl/Jn2KYi

François **Tonic**

Apprenons à programmer aux enfants

Un pari un peu fou : enfermer 30 enfants de 8 à 15 ans dans une salle informatique un mercredi après-midi ensoleillé pour leur apprendre à programmer (http://www.mix-it.fr/mixit14/mixteen). Alors ? Que s'est-il passé ? Les enfants ont profité de l'occasion pour surfer sur Internet ? Les animateurs ont déclaré forfait et organisé un foot ? Il y a eu indigestion de bonbons ?



Contre toute attente, les enfants sont rapidement devenus des apprentis développeurs, certains délaissant même le goûter pour corriger leurs bugs. Quand on devient geek, on n'a pas le temps de s'arrêter pour manger une crêpe !
Avec des logiciels adaptés comme Scratch (http://scratch.mit.edu/) ou Algoid (http://www.algoid.net/) pour les plus grands, ces enfants ont appris les bases de la programmation.
Les codeurs en herbe manipulent des objets, sur lesquels sont appliquées des méthodes. Oui, il s'agit de tortues qui se déplacent ou de personnages qui dansent. Mais les concepts sont là. Quiconque a tenté d'expliquer la programmation orientée objet à un non informaticien sait combien cette notion peut être difficile à acquérir.
Suite d'instructions, interactions avec le clavier, déclarations impératives comme les boucles, if then else, variables, et même objets ... Nous parlons bien d'algorithmes ! Ces notions sont seulement mises en pratique avec des outils éducatifs destinés aux plus jeunes.

Focus sur scratch

Le principe est classique pour un logiciel de programmation : diriger un objet en créant un script. L'objet est par défaut un chat, mais peut être tout autre personnage, à choisir parmi les bibliothèques d'objets ou à dessiner.
Le script est généré en déplaçant des briques par drag and drop, chaque brique correspondant à une instruction.
Les briques sont nommées avec l'instruction qu'elles portent, avec des mots simples : attendre, avancer, dire «salut» ...
Il n'y a pas de langage compliqué à apprendre, il suffit à l'enfant de savoir lire.
Les briques sont regroupées par thème : le contrôle, le mouvement, l'apparence... Chaque thème a un code couleur, ce qui permet aux enfants de mieux se repérer dans les instructions qu'ils génèrent.
Le thème opérateurs permet aux « matheux » de lancer des calculs, de comparer des nombres, d'obtenir un nombre aléatoire : pourquoi ne pas suivre le programme scolaire en jouant avec un chat ?
La compilation du programme est effectuée en cliquant sur un drapeau vert : enfantin!
La méthode des briques est vite assimilée. Les enfants veulent rapidement aller plus loin dans cet apprentissage qu'ils considèrent comme un jeu.
Une fois le premier objet défini et programmé avec quelques briques, les enfants peuvent programmer plusieurs objets et les faire interagir entre eux, comme dans un véritable programme.
Par défaut, leurs objets s'appellent objet1, objet2... Ne serait-il pas plus facile de les repérer entre eux s'ils avaient un nom plus significatif ? Évidemment. Ils comprennent tout de suite l'intérêt de renommer les objets avec un sens pertinent, principe qui n'est pas toujours respecté chez les informaticiens. Soyons réalistes, certains programmes professionnels contiennent parfois des variables appelées variable1, des procédures maprocédure ...
Pour enrichir les comportements de ces objets, des sons peuvent être ajoutés : leur propre voix, des sons pré-enregistrés de véhicules, d'animaux ... Les arrières plans sont aussi adaptables en fonction de l'envie de chacun.
Les objets peuvent avoir différents costumes, à personnaliser et à choisir par programmation.
Bref, tout est là pour que chacun laisse aller sa créativité, imagine son programme, tout en respectant la logique informatique.

Le bilan de l'après-midi

Les enfants ont réalisé des mini-scénarios de jeux (un requin mange des petits poissons et un score s'incrémente), des histoires un peu loufoques (des abeilles se volent du pollen de fleurs) ...
En quelques heures, ils ont compris comment fonctionnait un programme, et ont assimilé un vocabulaire informatique, sans forcément avoir de compétences initiales en développement.
Ils ont découvert la logique numérique, ce qui se cache à l'intérieur de ces boîtes noires que sont les ordinateurs de l'école, les tablettes, et autres smartphones des parents.
A la question «l'âge de votre premier programme informatique ?», ils peuvent maintenant répondre fièrement «8 ans»!

Séance d'Algoid

Avec cet atelier, les enfants expérimentent la programmation classique. Il ne s'agit plus de déplacer des briques, mais de taper de réelles lignes de code, de compiler, de débugger, d'exécuter, de comprendre, grâce au mode pas à pas, l'évolution du programme.
Et ça marche !
Les prérequis sont simples :

- Les enfants doivent avoir au moins 10 ans, ou être déjà un peu geeks dans l'âme.
- La connaissance des mots-clés en anglais (go, jump, turnleft ...) et des bases mathématiques (comment construire des formes géomé-

triques simples à partir d'un point de départ) permettent de débuter l'apprentissage.

Peut-être avez-vous tapé vos premiers programmes informatiques en langage Logo, et vous vous souvenez avec nostalgie de sa fameuse tortue. Et bien la tortue s'est modernisée, elle s'appelle Algo et a maintenant un look plutôt tendance.

La tortue Algo a des méthodes qui permettent de la guider :

- pour se déplacer:algo.go (100); algo.turnRight (90);
- changer de couleur: algo.setColor(3);
- parler: algo.text («hello algoid !»);
- disparaître/réapparaître: algo.hide(); / algo.show();.

Avec ces quelques instructions, il est possible de comprendre les bases de la programmation, pour réaliser ensuite des développements plus avancés.

Les tutoriels sont progressifs : dessiner d'abord un carré, puis un triangle, enchaîner par un escalier, et enfin réaliser une cible, ou d'autres figures géométriques, plus personnelles.

Grâce à ces exercices éducatifs, les enfants abordent la programmation en confiance et se prennent vite au jeu.

Ils apprennent dès le début à compiler, à exécuter le programme en mode pas à pas, à sauvegarder régulièrement leurs développements. Ne s'agit-il pas du B-A-BA de la programmation ?!

Ils comprennent aisément – pour ne pas dire naturellement – le principe des ordres donnés à la machine : la tortue ne réagit qu'aux instructions qu'elle connaît. Algo.joueauxbilles, ça ne marche pas, ou plutôt ça ne compile pas ... à moins de créer sa propre fonction. Une fois les premières notions acquises, effectivement ils arrivent à générer leurs fonctions (une fonction carre() qui se répète encore et encore pour former des graphiques amusants).

Ils utilisent des variables (le nombre de côtés d'un polygone), des boucles pour ne pas répéter la même instruction (loop), codent un «random» pour obtenir une figure aléatoire. Ils corrigent les bugs de leur programme : ce n'est pas vu comme une contrainte, mais plutôt comme la résolution d'une énigme.

Itératif, récursif : les mots ne seront peut-être pas retenus, mais les algorithmes codés utilisent ces notions, et les nouveaux formés les appréhendent avec facilité. Après avoir été initié à la programmation, ils volent de leurs propres ailes et trouvent presque eux-mêmes des sujets d'exercices :

- rechercher une valeur comprise entre 0 et 10, en faisant interagir l'utilisateur et l'ordinateur : «trop grand», «trop petit»
- coder un jeu de morpion : des carrés, des croix, des cercles, et c'est gagné

L'élève dépasserait-il le maître?

Mais Algoid a plus d'une corde à son arc : pour les plus téméraires, il offre toute une panoplie d'instructions pour construire des jeux et interagir avec l'utilisateur et son environnement.

Les conclusions de cet atelier

Au bout d'une après-midi, les enfants ont réussi à développer des programmes :

- soit orientés graphisme, avec de la recherche esthétique : pyramides, cercles imbriqués. Qui doute encore qu'on ne peut concilier l'art et l'informatique ?!
- soit logiques, codant de vrais algorithmes avec des interactions homme-machine.

Ces enfants ont maintenant les clés de la programmation et peuvent aller plus loin. Ils sont autonomes pour créer leur propre code, et ont compris les possibilités que pouvait leur offrir cette compétence informatique,

pour résoudre des problèmes mathématiques, inventer des jeux.

En quelques heures, ils ont appris un premier langage objet, et repartent tous avec l'envie de continuer cet apprentissage.

Nul doute qu'ils sauront s'adapter à d'autres langages, quand le moment sera venu, ou simplement demain pour le plaisir de découvrir de nouvelles façons de coder.

Et l'agilité dans tout ça?

Pour compléter cette expérience informatique, quelques principes agiles sont même appliqués :

- Pair programming : positionnés en binôme, les enfants s'aident et se motivent. Non, ils ne se battent pas pour tenir la souris. Les habitués: «c'est mon 4ème coding goûter» aident les novices, qui sont emballés : «C'est mieux qu'un anniversaire !». Ils partagent leurs idées et se guident l'un l'autre.
- Démonstration : à la fin de l'atelier, chaque enfant présente avec fierté le résultat de son travail aux autres. Le traditionnel «effet démo» est lui aussi présent : une clé usb vide, un essai sur un autre PC qui plante le programme. C'est comme ça aussi dans la vraie vie des informaticiens.

La démonstration devant le public averti du Mix-IT (http://www.mix-it.fr/mixit14) permet à quelques ingénieurs de se rappeler leur premier cours d'informatique : «Ils ont codé un jeu de morpion, j'ai appris ça à 20 ans ...»

- Amélioration continue : certains profitent des quelques minutes restantes pour améliorer leur code, en fonction de ce qu'ils ont vu pendant les démonstrations et des remarques de chacun. Tous demandent à avoir le code de leurs camarades, pour étoffer et diversifier leurs propres programmes.
- Refactoring : les plus avancés se sont lancés dans les fonctions. Dupliquer du code, pas question ! Ils comprennent l'intérêt de factoriser leurs programmes de façon presque naturelle. Avec des variables, des boucles, des fonctions, il est possible de simplifier et de pérenniser son code, plutôt que de copier/coller la même instruction. Mais oui, ils relisent leur code, sans rechigner !

Quel avenir pour les TeenCoders?

Apprendre l'informatique aux enfants, très bonne idée, mais cette matière n'est-elle pas déjà enseignée en primaire? Oui, mais ...

Un brevet informatique et internet (Le B2I http://eduscol.education.fr/cid46073/b2i.html) est mis en place depuis la rentrée 2012 pour attester des compétences informatiques des écoliers, collégiens et lycéens. Alors, qu'apprennent les écoliers pendant ces cours d'informatique?

Comment utiliser un clavier, une souris, rechercher sur Internet, rédiger des documents, écrire un blog sur les activités de la classe ... Indispensable dans notre monde de technologies.

Programmer, non, pas vraiment.

Les initiatives de programmation pour les enfants se multiplient donc: Programatoo, Devoxx4Kids (http://www.devoxx4kids.org/france/), coding goûters partout en France...

Les organisateurs de ces sessions ont compris que la programmation était accessible à des enfants, avec des logiciels adaptés, et que l'école ne dispensait pas encore ces formations. Les parents ont souvent un pied dans l'informatique. Passionnés ou non de programmation, ils veulent proposer aux enfants cet apprentissage : pour mieux comprendre l'outil informatique qui sera leur quotidien, et surtout le maîtriser.

Les enfants quant à eux viennent là pour s'amuser, découvrir un autre univers, pas forcément pour travailler... Curieux de tout, ils apprennent les concepts fondamentaux de la programmation de façon ludique. Souvent, ils continuent l'expérience chez eux et sont prêts à revenir coder avec plaisir. Programmer est un jeu, pas besoin d'attendre d'être grand pour en profiter. Chaque enfant est capable de comprendre le numérique, de créer ses propres programmes et pas uniquement de consommer l'outil informatique. Avec un ordinateur, quelques d'heures de formation sur des logiciels gratuits, des copains avec qui partager l'expérience, l'affaire est entendue : chacun peut devenir programmeur. Développeurs en tout genre attention : la relève est en marche, prête à tout coder !

Aude **LEMAR-VERRIER**
Assistance à Maitrise d'Ouvrage - VISEO

Doit-on enseigner le code aux enfants ?

On peut facilement répondre par un oui ou par un non à cette question, mais si on veut creuser un peu plus, on se sent obligé de se poser d'autres questions afin de mieux cerner la question, sinon la réponse. En voici quelques-unes qui ont traversé mon esprit.

Doit-on enseigner aux enfants le jardinage ? Doit-on enseigner aux enfants le bricolage ? Qui veut enseigner le codage aux enfants et quels sont leurs arguments ? Si on a envie d'apprendre à coder, l'école est-elle le meilleur moyen d'y parvenir ?

Il y en a plein d'autres nécessairement, mais j'espère, en vous présentant mon lot de subjectivité, vous aider à choisir vous-même si vous voulez que votre enfant sache coder.

Doit-on enseigner aux enfants le jardinage ?

Vous vous demandez quel rapport ? Il y en a un, un peu lointain je vous l'accorde, mais quand il s'agit d'enfants et d'avenir, mieux vaut ne pas avoir la vision courte.

A OOPSLA 97, Alan Kay, lauréat du prix Turing, dans son discours « The Computer Revolution Hasn't Happened Yet », décrivait une vision de l'informatique et des systèmes d'informations qui « pousseraient » comme des plantes ; l'informaticien aurait donc plus un rôle de jardinier que de codeur. Personnellement, je prends très au sérieux Alan Kay, à tel point que j'ai cofondé la société Aspectize en 2007 pour aller modestement dans son sens.

Il est difficile de faire pousser un TGV : le métal inerte, les dizaines de milliers de pièces taillées sur mesure et emboitées les unes dans les autres font sa « rigidité » et l'empêchent de pouvoir pousser ; alors que les plantes ou les crabes utilisent des mécanismes qui leur permettent de changer, de s'adapter et de croître sans qu'ils soient obligés d'être arrêtés pour la maintenance.

La souplesse naturelle du monde vivant a peut être, dans une certaine mesure, déjà été transposée au monde du logiciel : on a réussi à séparer le matériel (ordinateur, processeur, câble et antenne réseau) du logiciel, cette chose immatérielle, sans masse, sans longueur ni largeur, ne respectant pas les lois physiques. Cette séparation ouvre un champ d'action nouveau, nécessitant peut-être une mathématique et une physique nouvelles. Ecrire des logiciels adaptables est quelque chose de possible. Internet est un bon exemple de système qui pousse.

Doit-on enseigner aux enfants le bricolage ?

J'emploie ici le terme « bricolage » au sens péjoratif. Les 80 premières années de l'informatique ont été marquées par des batailles commerciales, des « moi aussi » et une imagination trop colonisée par l'approche industrielle de fabrication de pièces rigides. Les systèmes informatiques en place aujourd'hui sont construits de bric et de broc sur le champ de ses batailles-là. Et les informaticiens, chacun barricadé dans son champ de spécialité, de brandir à son voisin le micro-avantage de son code, son langage ou ses outils (il y en a des centaines).
Le code pose beaucoup de problèmes et connait encore des évolutions importantes. Contrairement au français ou à l'anglais qui sont faciles à lire mais difficiles à écrire, le code s'écrit facilement et se lit très difficilement, ce qui fait que chaque équipe d'informaticiens arrivant sur un projet propose de démanteler le travail de la précédente, faisant exploser ainsi les coûts et les délais des projets. Entre 60 à 70 % des projets informatiques sont considérés en échec par le rapport Chaos publié régulièrement depuis 1985 par le Standish Group.

Vous connaissez certainement les expériences de Konrad Lorentz sur l'attachement. A leur naissance, les bébés canards prennent le premier être vivant qu'ils voient pour leur mère et commencent à le suivre. Cela met une responsabilité importante sur les épaules de ce premier être vivant, parce que si au lieu d'entrer dans l'eau il commence par grimper sur un arbre, il induira implicitement chez les canetons la frustration d'être incapables de vivre en harmonie avec leur environnement.
Si le premier contact de l'enfant avec l'informatique est cet aspect « bricolage » du codage, on risque de créer une génération qui maîtrise mal son sujet.

Qui veut enseigner le codage aux enfants et quels sont leurs arguments ?

En cherchant les arguments « pour », je suis tombé sur deux catégories de personnes : des informaticiens célèbres qui ont réussi et des hommes politiques, présidents, ministres et députés.

Quels sont les arguments des informaticiens qui ont réussi ?

Steve Jobs – davantage visionnaire qu'informaticien - : « I think everybody in this country should learn how to program a computer because it teaches you how to think. ». De la carte perforée au Javascript, j'ai codé quasi quotidiennement depuis plus de trente ans et je continue de le faire. Voici mes réflexions sur cette phrase.

Le prémisse est juste : coder apprend à réfléchir puisque que tout ce qui nécessite réflexion nous apprend à réfléchir, et coder n'en est qu'un exempte. Par contre la conclusion - tout le monde doit coder - me paraît suspecte ! Ceux qui aiment coder peuvent le faire, ceux qui n'aiment pas ont intérêt à aiguiser leur réflexion avec une autre activité qui leur conviendra mieux.
Si la phrase de Jobs était juste (prémisse juste, conclusion juste), tout le monde devrait pratiquer toutes les activités qui nécessitent réflexion, des échecs au cambriolage d'une banque en passant par le Monopoly !

Coder apprend à réfléchir. Cela favorise une forme de réflexion qui permet à un humain de décomposer un problème en termes digestes pour une machine. Se pose donc une autre question : doit-on, veut-on, développer cette forme de réflexion pour tout le monde ?

Que disent les autres informaticiens ?

Un bref tour sur le site code.org vous familiarisera rapidement avec les arguments de nos informaticiens célèbres. On apprend par exemple que certains ont commencé très tôt - à 8 ou 13 ans -, que coder était amusant pour eux, que tout le monde pouvait le faire. On découvre aussi que coder est le bon choix pour gagner de l'argent, ou pour « changer le monde ».

Certains regrettent de ne pas avoir su plus tôt que « coder permet de créer des logiciels utiles aux hommes » ! D'autres sont émerveillés par le fait d'être « capables de diffuser leur travail au monde entier » ! D'autres cherchent à recruter en évoquant « les super pouvoirs magiques » que procure le codage à

celui qui consent à l'apprendre ! Etant donné le manque d'informaticiens, on entend aussi que pour les attirer il faut créer des lieux de travail sympas et agréables (possibilité de jouer de la musique, possibilité de faire du sport, nourriture, lieux de convivialité, service de blanchisserie...).

Il est intéressant de relever que tous nos témoins ont réussi grâce à leur passion du codage - en dehors de l'école. Qu'est ce qui les pousse à vouloir voler le plaisir de la découverte et de l'autonomie à leurs enfants, en transformant une activité intéressante pour certains en une activité scolaire pour tous ? Un acteur de cinéma qui a commencé jeune pense-t-il que le cinéma doit être enseigné à l'école primaire ? Un humoriste qui diffuse ses sketches sur Youtube demande-t-il que l'humour soit enseigné à l'école ?

On peut se demander pourquoi d'un côté on aguiche le codeur en lui offrant blanchisserie et la nourriture et de l'autre on délocalise son travail vers des contrées lointaines ? L'engouement démontré pour le codage sur le site code.org s'explique par l'offre et la demande : peu de codeurs et une demande élevée. Cet enthousiasme s'estompera quand l'offre et la demande s'équilibreront.

Quels sont les arguments des dirigeants politiques ?

En France pour le moment on parle d'activités facultatives sur le temps périscolaire (TAP) ; c'est-à-dire dans la même catégorie que le théâtre, l'art contemporain, l'escrime, les échecs, la réalisation de fresques... Pourquoi pas, si un enfant s'amuse plus en codant qu'en faisant de l'escrime, alléluia ! Il n'y a pas de quoi fouetter un chat.

Mais parallèlement, on entend des phrases comme : « On apprend bien l'anglais et l'allemand, pourquoi pas le HTML » (!) ou « Le code, le nouveau latin », ou encore : « Les objectifs fondamentaux et prioritaires qui doivent être assignés aux écoles sont l'apprentissage de la langue française, la maîtrise de la lecture, de l'écriture, l'utilisation des mathématiques et l'apprentissage du codage informatique » qui contribuent à la confusion sur la question.

Dire qu'on apprend « l'anglais pourquoi pas le HTML » ressemble à : « On mange bien des canards pourquoi pas des chauves-souris ». Ce n'est pas parce que le L de HTML veut dire Language qu'il faut le confondre avec une langue vivante comme l'anglais ou une langue ancienne comme le latin ! Communiquer avec ses semblables n'est pas la même chose que programmer une machine !

Mettre au même niveau lecture, mathématiques et codage est assez curieux aussi. Lire et écrire sont des activités qu'on pratique durant toute sa vie. Et les notions mathématiques, certes souvent inutiles, ont une solidité millénaire. On vit très bien sans savoir résoudre une équation de second degré, mais aucun prof de math au monde ne va mettre en question la solution, alors que si on demande à 100 informaticiens d'écrire un programme qui justement résout des équations de second degré, on aura 100 programmes différents ; je parie que dans le lot certains ne marcheront pas toujours. Le codage informatique n'a rien de fondamental ni pérenne.
Dans une vidéo, on voit le président américain demander aux jeunes de son pays de s'engager dans l'informatique pour l'avenir d'eux-mêmes et de leur pays. Il leur demande d'au lieu d'acheter un jeu vidéo d'en développer un, au lieu de télécharger une application d'aider à sa conception, et enfin au lieu de jouer sur son téléphone de le programmer.

Ce qui me plaît dans ce discours, c'est qu'il parle d'engagement et pas de cours scolaire, il parle aussi de conception informatique et pas seulement de « codage ». Mais il est vain de demander à quelqu'un qui joue à un jeu de s'intéresser aussi à son développement : beaucoup d'informaticiens consomment des jeux vidéo alors qu'ils ne seraient pas capables d'en développer un. On ne demande pas au spectateur d'un film de faire du cinéma ni à un conducteur de voiture d'en fabriquer une !

L'informatique est-elle en train de changer le monde ?

L'ère numérique a démarré il y a quelques millénaires avec Pythagore qui disait que tout était nombre. Techniquement et socialement on est en train de lui donner raison. Grâce aux ordinateurs fixes mais surtout mobiles, internet, et de plus en plus avec les objets communicants, le numérique est en train de changer notre monde, notre vie, nos relations avec les institutions et les uns avec les autres. De nos jours, quand on va en tête à tête au restaurant, on invite plein d'autres personnes à table à travers notre vie réticulaire. La puissance de calcul et de tri de nos machines ont rendu l'espionnage possible à une échelle planétaire. On peut commander une voiture avec chauffeur ou réserver une place de parking à tarif imbattable à travers des systèmes d'information mis à notre disposition à travers deux ou trois clics sur nos téléphones... Depuis l'invention du livre, tous les « progrès » en matière de communication allaient dans le sens des communications de 1 vers N (un auteur/N lecteurs, une émission/N téléspectateurs ou auditeurs). Ce côté « tyrannique » a été rompu par internet, le web, les ordinateurs mobiles et les technologies de l'informatique moderne.

Ce champ de liberté nouveau a provoqué l'explosion de la demande en informatique et en informaticiens et on se trouve subitement en manque de professionnels capables de parler aux machines.

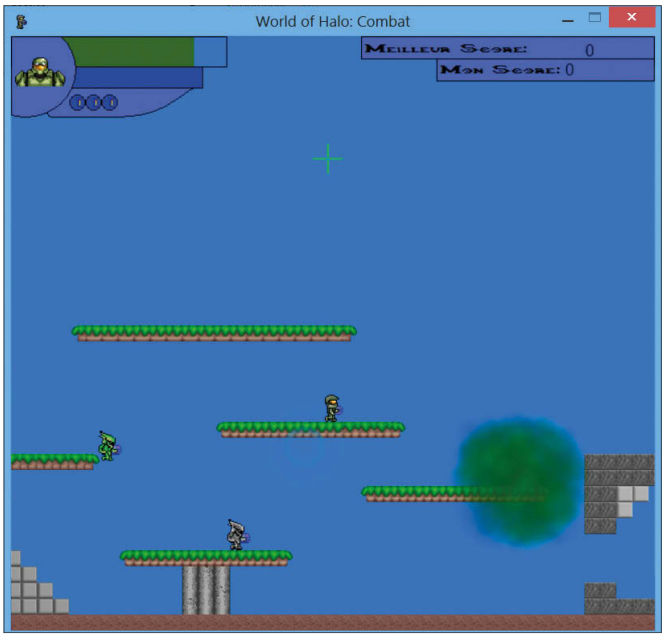
Le travail de l'informaticien consiste à orchestrer la circulation de l'information entre les hommes et les machines. Concevoir la forme et le flux de cette information, imaginer les interfaces avec les humains pour que le contrôle du flux soit en conformité avec leurs besoins, choisir les outils et techniques à utiliser pour que tout cela fonctionne de manière stable et prévisible, voilà le besoin, voilà le travail de l'informaticien : tout cela ne fait pas intervenir du code, c'est avant tout un travail intellectuel. Mais pour que tout cela marche, il faut écrire du code – et moins on en écrit, mieux on se porte.

L'informatique est destinée à perdurer, nous sommes en train de vivre le début de son commencement, alors que le « codage » vit le début de sa fin. Enseigner à l'école un métier du futur avec des techniques du passé ne sert pas ce métier.
Personnellement je ne pense pas que l'école ait un impact positif sur l'apprentissage. On apprend par intérêt, on apprend en s'amusant, on apprend quand on a envie, avec le rythme et l'énergie qu'on a à consacrer à un sujet. C'est comme cela qu'un bébé apprend à parler et à marcher de façon très efficace, animé par sa motivation d'imiter ses semblables.
On fera un pas en avant si on arrive, à travers l'informatique, internet, l'interaction entre les uns et les autres, à créer des espaces réels ou virtuels d'apprentissage. On pourrait y apprendre le code mais aussi l'histoire, l'anthropologie etc., de telle sorte que chacun puisse à son rythme, selon les sujets et les moments qu'il a choisis, construire son équilibre avec le reste de la société, sans jugements, sans notes, sans promesses de diplômes et de réussite.

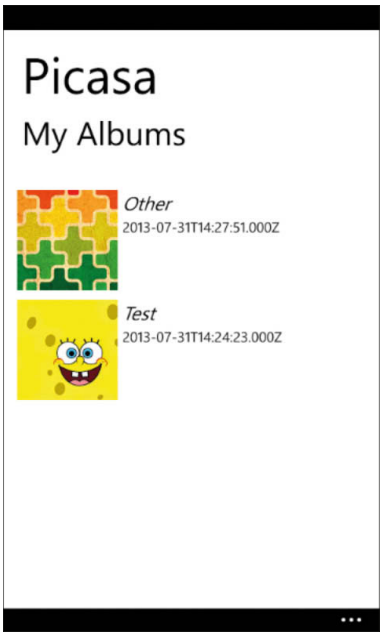
Frédéric **Fadel**

Développeur à 12 ans

Je suis un jeune étudiant de 19 ans qui développe depuis l'âge de 12 ans. Je prépare pour le moment un BTS SIO, je viens tout juste de terminer la première année.



(World of Halo – Jeu PC réalisé avec Game Maker)



(Picasa pour Windows Phone – Application non officielle)

J'ai, depuis ma petite enfance, été attiré par les ordinateurs et machines informatiques en tous genres, qui semblaient pouvoir faire des choses incroyables. Et comme quoi le hasard fait parfois bien les choses, j'ai à la maison une personne qui parle couramment l'ordinateur : mon père, Stéphane Sibué.
Je l'ai régulièrement vu développer sur notre vieil ordinateur de l'époque, où il réalisait divers programmes plus compliqués les uns que les autres pour ses clients. Et c'est déjà à ce moment-là que la petite étincelle du développeur est née en moi. Et je me posais mille questions : « 'comment font-ils cela ? »', « 'comment est-ce que ça fonctionne ? »' ou encore, et c'est là le plus important : « 'comment puis-je faire la même chose ? »'.

C'est alors qu'à 12 ans, j'ai trouvé sur le NET un petit environnement de développement qui permettait sans aucune notion de programmation, de développer des jeux 2D et 3D : Game Maker. J'ai donc tout naturellement commencé à creuser le sujet, dans le but bien évidemment de créer un premier jeu et accessoirement pour épater la galerie.
J'ai donc pendant plusieurs années créé des jeux plus ou moins compliqués, certains en 2D d'autres en 3D, en réseaux. Mais c'est amusant quelque temps de faire des jeux, je voulais créer de véritables applications pour ordinateurs.

J'ai donc commencé à apprendre le langage VB.NET avec Visual Studio 2008. Le début ne fut pas simple. La notion d'objet et de classe fut pour moi particulièrement difficile à comprendre, même si une fois la chose faite, on ne peut plus s'en passer. J'ai commencé le développement orienté objet. J'ai eu la chance, tout au long de mon apprentissage, de pouvoir poser mes questions à mon père et ainsi obtenir des réponses qui m'ont permis d'avancer rapidement. Ces langages ayant déjà un certain âge, les différentes communautés sur internet m'ont aussi beaucoup aidé.

Je maîtrise aujourd'hui divers langages comme C#, PHP, JavaScript, VB.NET, Java. Et bien entendu sur plusieurs plateformes comme Windows Phone, Android, Desktop (application classique) et Web.

Mais il est vrai que commencer seul dans cette discipline, sans cours comme on aurait pu les avoir à l'école, est parfois handicapant. Puisque nous n'avons pas de leçon sur lesquels nous appuyer lorsqu'on a un trou de mémoire, pas de professeur (hormis dans mon cas, mon père), à qui poser nos questions.

C'est pour cela que je pense qu'apprendre la programmation à l'école, ou tout du moins la notion de développement avec sa logique, peut être bénéfique à tous, même pour la vie de tous les jours. Régulièrement, beaucoup de personnes et de tous âges me demandent un coup de main parce qu'ils ne savent pas utiliser leur(s) ordinateur(s) alors que nous en avons de plus en plus besoin, notamment dans la vie scolaire, professionnelle, et même personnelle qui tourne dorénavant beaucoup autour des réseaux sociaux et autres flux d'informations communautaires.
Il serait donc bon d'enseigner aux plus jeunes l'informatique dans un sens large afin qu'ils aient des notions et ne soient pas perdus plus tard.

Mais cela sous-entendrait de revoir l'ensemble des formations actuellement disponibles, comme notamment le BTS que je fais. Puisque celui-ci est conçu pour apprendre de A à Z les bases de l'informatique. Ce qui pour ceux qui ont déjà un peu d'expérience peut devenir pénible de revoir sans cesse les mêmes choses.

Cela pourrait permettre aussi à beaucoup d'enfants de découvrir toute une gamme de métiers qu'ils ne connaissent pas, et qu'ils pourraient alors envisager comme futur métier. Sachant que ce domaine est de plus en plus présent dans notre économie.

Je pense que cela permettra également à la plupart des gens qui nous entourent et qui ne comprennent pas forcément ce qu'est l'informatique, d'être plus familiarisés. Puisque parfois, nous ne sommes pas bien compris ou bien vus par les autres. Certains nous voient comme des « 'bidouilleurs »' ou hackers et d'autres ne comprennent tout simplement pas. Néanmoins cela ne doit pas vous décourager de découvrir l'informatique, et notamment le développement puisqu'il s'agit là, d'un très beau métier de créativité et d'inventivité.

Kévin **Sibué**
http://www.kevinsibue.com
ksibue@gmail.com

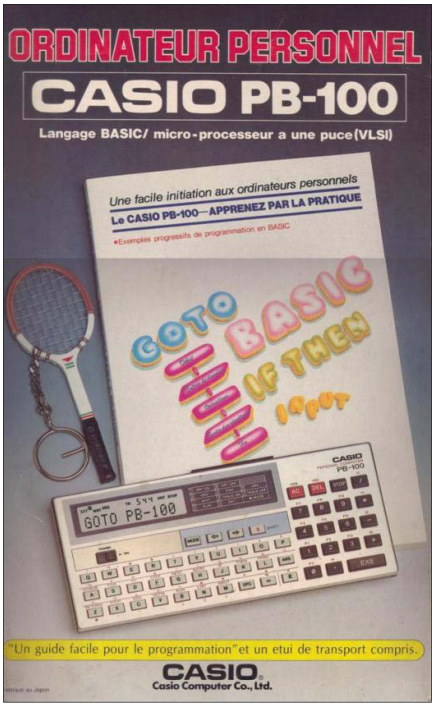
Pourquoi apprendre la programmation aux enfants ?

Je suis développeur de métier depuis plus de 20 ans et cela fait plus de 30 ans que je programme. J'ai commencé à 12 ans, en 1980. A cette époque les choses n'ont pas été simples.

Tout a commencé quand mon père m'offrit pour Noël une console de jeux Philips Videopac. Au contact de cette machine très limitée je me suis mis à rêver de pouvoir programmer des jeux. Quelques mois plus tard, Philips mit en vente une cartouche qui permettait de réaliser des programmes en assembleur. Pas super simple d'accès pour un débutant, d'ailleurs le manuel expliquant comment faire était en anglais ! Ce ne fut pas simple de comprendre ce qu'était un accumulateur, un registre, un flag, et tous ces termes propres au développement en langage machine. J'ai fait preuve à l'époque d'une sacrée dose de persévérance pour arriver à sortir quelque chose des tripes de ma console.

Cartouche programmation Vidéopac

J'ai voulu aller plus loin. Au CDI de mon collège j'ai trouvé quelques livres qui parlaient du Basic, et du monde des ordinateurs. Apprendre le Basic dans des livres c'est bien, mais très vite le manque de matériel permettant de mettre en pratique s'est fait sentir. Mon père voyant que cette nouvelle lubie de programmer était sérieuse m'a alors acheté un Casio PB 100 programmable en Basic et là j'ai pu avancer. Je crois d'ailleurs que ma passion pour la mobilité est née de la programmation de ces petites machines qui tenaient dans la poche et qui permettaient, pour l'époque, de faire beaucoup de choses. Les choses se sont ensuite emballées avec l'arrivée de l'Amstrad CPC 464, puis un PC, ce qui m'a permis de découvrir d'autres langages tels que le Pascal, le C, le Logo. J'ai fait de ma passion mon métier et c'est toujours avec autant de joie que je code. J'ai une grande préférence pour le développement mobile et je dois bien avouer que je me régale à coder des applications pour iOS, Android ou Windows Phone.



Casio PB-100 et son livre sur le Basic

J'ai fait tout ce chemin seul, personne dans mon entourage n'était capable de m'aider. J'étais une sorte d'extraterrestre aux yeux de mes copains qui ne comprenaient pas comment je pouvais préférer rester tout l'après-midi à coder plutôt que d'aller avec eux à la piscine courir après les filles !

Je regrette que la programmation ne soit pas étudiée à l'école, et ce dès le plus jeune âge car elle m'a permis d'avoir un bon esprit d'analyse et une logique dans la résolution de problèmes. J'ai toujours été nul en maths et pourtant je suis un bon développeur (du moins c'est ce qu'en disent mes clients). A l'école on apprend la logique par les maths, alors que ce n'est pas forcément la meilleure manière de faire, la programmation étant une très bonne alternative. Je vois beaucoup de personnes qui sont esclaves de leur ordinateur. Elles ne savent pas comment il fonctionne, alors elles utilisent les quelques petites choses qu'elles ont apprises çà et là sans jamais comprendre comment éviter certains problèmes. Si on apprenait aux enfants comment fonctionne un ordinateur, on créerait une génération maitresse et non esclave de ces machines, et pas mal de vocations naîtraient plus tôt pour l'informatique et la programmation.
Mon fils a suivi le même parcours que moi et programme depuis ses 12 ans. La différence c'est que j'étais là. Il a toujours voulu se débrouiller tout seul le plus possible, mais quand il était vraiment coincé j'étais là pour l'aider, le guider, et lui éviter de s'épuiser à trouver des réponses à des problèmes trop ardus.
Actuellement il prépare un BTS SIO (programmation), et comme à mon époque (même BTS passé en 1988) il s'est retrouvé avec une majorité d'élèves qui n'avait jamais vu une ligne de code et ne savait pas comment fonctionne un ordinateur. Ils se sont lancés dans une carrière de développeur sans aucune notion scolaire. Je peux vous dire que mon fils n'a pas trouvé cette 1ère année super intéressante, il a passé son temps à revoir ce qu'il sait déjà depuis bien longtemps.
A bien y réfléchir, l'enseignement de la programmation dès l'école primaire pourrait être d'une très grande utilité et permettre aux enfants d'éveiller leur esprit très tôt et faire naître de belles vocations, et à notre pays d'être finalement plus fort face aux enjeux qui se dessinent pour les 30 prochaines années et qui sont indéniablement liés à l'informatique sous toutes ses formes.

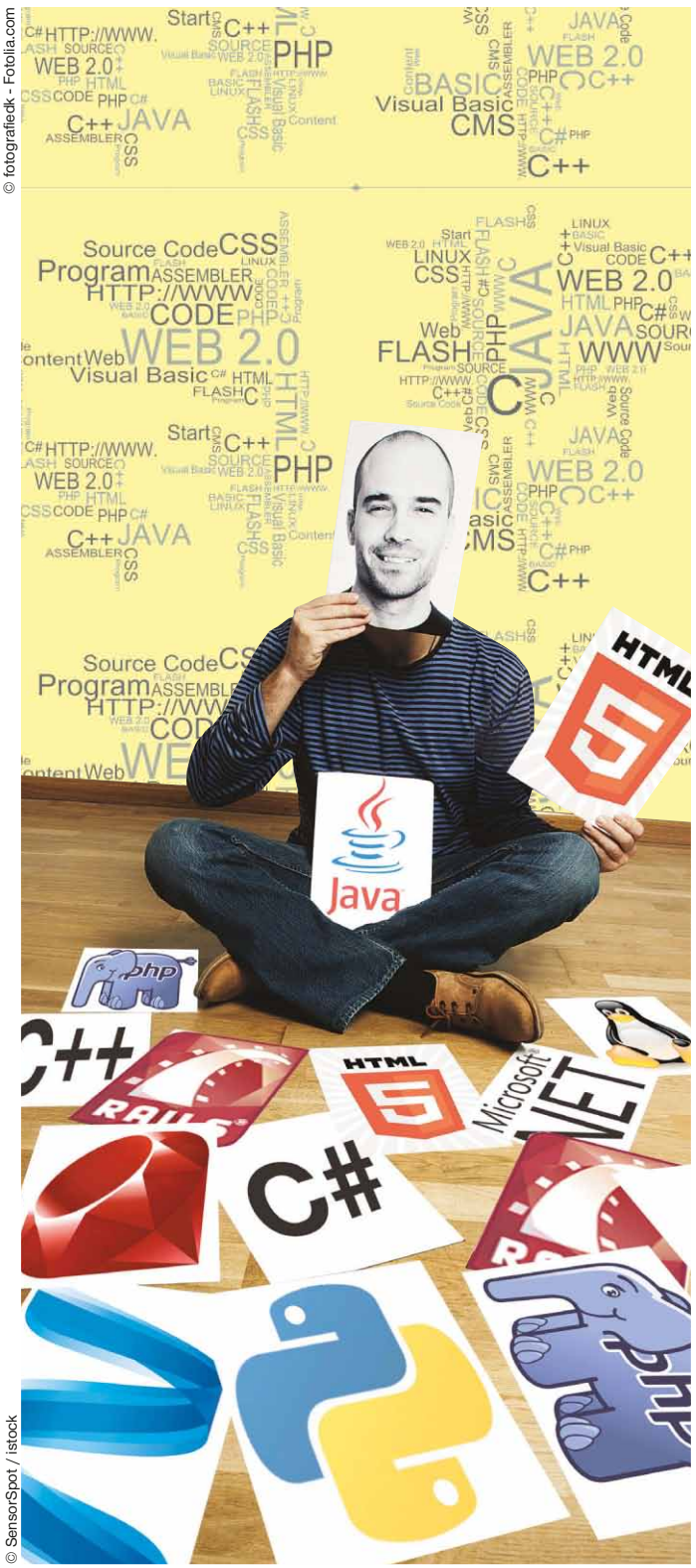
Stéphane **Sibué**
Directeur technique chez GUYZMO
stephane.sibue@guyzmo.fr
www.guyzmo.fr

GUYZMO

Comment choisir son langage de programmation ?

Le choix d'un langage de développement a toujours été une question délicate et la multiplication des langages n'aide pas. Aujourd'hui, nous trouvons des langages pour tout et n'importe quoi. Il n'y a qu'à regarder l'index Tiobe.

Mais finalement, souvent, le langage s'impose de lui-même : à l'école, dans un projet, en entreprise. Cependant, ce choix imposé n'est pas forcément le plus pertinent. Il est encore trop souvent dicté par la sécurité et les habitudes. Ce n'est pas parce qu'un langage est populaire qu'il est la bonne solution. Théoriquement, le langage doit être choisi selon une méthodologie pragmatique et neutre. C# ou Java n'est pas la réponse à tout, ni C++ ou JavaScript ou PHP. Les index de popularité de type Tiobe sont à manipuler avec prudence...



© fotografiedk - Fotolia.com

© SensorSpot / istock

La pertinence d'un langage dépend de nombreux critères/éléments : communauté, outils disponibles/compatibles. À cela se rajoutent : les évolutions du langage, la syntaxe, les performances, la maintenance du langage.

Surtout, aujourd'hui, un développeur ne doit pas se limiter à un langage. Il doit absolument en connaître et en pratiquer plusieurs, et être capable d'en apprendre un nouveau très rapidement. Le développeur polyglotte est une réalité. Et il doit regarder ce qu'il se passe au-delà de ses langages de prédilection.

La veille technologique sur les langages sera un élément important pour savoir ce qu'il se passe...

François Tonic

Comment choisir ses langages de programmation ?

De l'Assembleur, Fortran et COBOL des années 1950 au Swift de 2014, l'informatique a connu plusieurs milliers de langages de programmation. La plupart ne sont connus que par leur créateur – qui ne l'utilise pas. Certains comme Whitespace sont dans la catégorie humour. D'autres sont célèbres et irremplaçables dans certains domaines, beaucoup connaissent des hauts et des bas en fonction de la mode du jour.

Beaucoup de langages sont utiles et utilisés
Il existe de nombreux critères pour choisir ses langages de programmation. Il y a les critères liés aux goûts, aux croyances, aux compétences et à l'ego du développeur : un freelance en Californie a bien plus intérêt à connaitre Python que Java puisque la compétence est plus rare et la demande est élevée. Alors qu'en France, à part pour briller dans les conversations de geek, la connaissance de Python n'est pas si demandée.
Egalement, les contraintes techniques et environnementales peuvent influencer le choix d'un langage. Par exemple si vous devez développer dans le domaine du temps réel, un langage avec gestion automatique de mémoire (Garbage Collector), du style C# ou Java, ne conviendrait pas. Ou bien si vous avez des besoins de tolérance aux pannes et/ou de parallélisme massif, vous vous tournerez vers Erlang.
Personnellement j'aimerais dire que peu importe le langage, l'important dans notre métier, qui est en train de changer le monde, c'est l'utilité, la qualité, le coût et la rapidité du développement logiciel. La valeur d'un logiciel qui vous permet de trouver une place de parking - qui vous convient, loin de chez vous - ne réside pas dans le langage de programmation utilisé, qu'il soit Ruby ou C#.
Néanmoins, il y a des critères plus ou moins objectifs qui peuvent influencer le choix du langage de programmation. Voici un florilège de critères et de recommandations - forcément biaisées -, qui peuvent vous inspirer.
D'abord le « fun » : programmation ou jardinage, si le travail se fait sans amusement, il ne sera pas très productif. Maintenant, on peut s'amuser de différentes manières : si ça vous amuse d'être le seul à maîtriser un hiéroglyphe que d'autres ne comprennent pas, je vous recommande APL. Cela dit, on peut faire des sacs de nœuds avec tout type de langage.
Si les accolades vous perturbent, vous pouvez choisir un langage ayant une syntaxe type Python, qui n'a pas que cet avantage. Si vous avez tendance à confondre les majuscules et les minuscules, optez pour un langage type VB.
Après le « fun », on peut considérer la performance. Ici, le débat entre ceux qui comptent les nano secondes et ceux qui optimisent globalement, ceux qui préfèrent la compilation statique ou la compilation dynamique voire l'interprétation est sans fin, et souvent sans intérêt.
Si vous êtes un fanatique de la nano seconde, vous opterez pour l'assembleur ou le C -définitivement. Mais n'oubliez pas que la performance d'une application ou d'un système d'information est rarement due au choix du langage de programmation.
Maintenant il est clair que certaines opérations sur un iPhone ne sont pas assez performantes en Javascript, et Objective C sera alors nécessaire, mais ce manque de fluidité est dû essentiellement à l'architecture matérielle et logicielle de l'iPhone, et pas à l'aspect dynamique ou interprété du Javascript.

Impératif, déclaratif et les autres styles

On peut aussi être adepte d'un style de programmation particulier : Il y a le style impératif, le style déclaratif ou encore l'orienté aspects.
Ces styles de programmation ont été influencés par le lambda calcul d'Alonzo Church et la machine d'Alan Turing. Ces deux approches ont été mises au point pour répondre à la question (Entscheidungsproblem, 1928) de David Hilbert quant à l'existence d'une procédure uniforme de décidabilité en mathématique. Turing (1935) et Church (1936) ont répondu négativement à la question.
La machine de Turing a donné naissance aux langages - ou style - de programmation impérative. Dans ces langages, le développeur décrit minutieusement des instructions à exécuter par la machine pour obtenir le résultat voulu.
COBOL, FORTRAN, C, PHP et toute la famille des langages dits « orienté objet » C++, Java, C#, VB.NET, font partie de la catégorie des langages impératifs.
Le lambda calcul a donné naissance aux langages - ou style - de programmation déclarative, dont font partie les langages fonctionnels. Ces langages sont généralement plus concis et la forme d'abstraction qu'ils favorisent est plus mathématique. Cette approche fonctionnelle - contrairement aux pratiques « orienté objet » - est assise sur des théories mathématiques solides, ce qui fait son charme pour certains et sa difficulté pour d'autres. Ici aussi il y a débat entre les puristes, qui collent à la théorie, et les pragmatiques qui se permettent des écarts de temps en temps. Parmi les langages qui permettent ce style, il y a : Lisp (1958), Scala (2004), Haskell (pour les puristes, 1990), Python (1989), F# (2007), Erlang (1986) et de plus en plus le C# qui est néanmoins plus connu pour son caractère impératif.
Si écrire le minimum de code nécessaire pour une fonctionnalité, une appli ou un système d'information distribué vous anime, je vous recommanderais de vous familiariser avec un langage fonctionnel ou du moins avec l'approche déclarative. Les fanatiques de l'orienté objet ne s'amuseront pas assez avec les langages fonctionnels, néanmoins je les invite à s'y frotter, car même s'ils ne vont pas adopter l'approche, ils trouveront plein d'idées et de pratiques intéressantes et instructives, utiles même en Javascript.
Comme style de programmation, il y a enfin l'AOP (Aspect Oriented Programming). Dans cette approche, l'accent est mis sur la séparation des aspects techniques et des aspects métier à travers des techniques dites d'inversion de contrôle (IoC). Bien qu'il y ait quelques langages spécifiques comme AspectJ et AspectC++, ce style de programmation - mon favori - peut être mis en œuvre à travers d'autres langages, le C# par exemple qui propose un certain nombre de constructions particulières facilitant cette approche.

Au-delà...

Au-delà du langage, il existe tout un environnement comprenant notamment des bibliothèques utilitaires : si vous avez besoin de calculer la racine carrée d'un nombre, ou de faire des entrées/sorties avec votre système de stockage ou d'affichage, il vous faut nécessairement une bibliothèque standard ou particuliè-

re pour le faire. Depuis l'ère .net, grâce à son système de types unique, tous les langages .net utilisent les mêmes bibliothèques d'utilitaires : votre calcul de racine carrée ou votre peinture de l'écran se fera de la même manière en C#, F# ou A#, alors qu'en C++ ou Ruby vous avez besoin de bibliothèques et de connaissances différentes.

A mon sens, ce seul fait donne un avantage majeur aux langages .net.

Dans le monde du calcul scientifique, l'existence d'un savoir-faire et de bibliothèques utilitaires en FORTRAN font que ce langage est pour le moment indétrônable dans sa niche.

En tant que développeurs, une bonne partie de notre temps consiste à écrire et mettre au point nos programmes. On a donc besoin d'un environnement de développement riche, vous choisirez donc plutôt un langage qui est livré avec un tel IDE.

Le choix du langage est également influencé par la plateforme d'exécution : web ou non, type de système d'exploitation, et de nos jours selon les différents AppStores comme ceux d'Apple, Google, Microsoft, Firefox, Chrome...

Si vous développez pour et avec les technologies web - on dirait HTML5 aujourd'hui -, vous n'échapperez pas au meilleur des langages, le Javascript, un langage dynamique, minimaliste, permettant un style de développement fonctionnel, mais aussi orienté objet si vous insistez. Ce langage a certes quelques défauts, mais la souplesse qu'il offre au développeur maître de son art est sans égale.

Pour le développeur web coté serveur, le choix du serveur est souvent déterminant. Si c'est un Linux, il y a des chances qu'il opte pour le PHP ou Ruby (c'est à la mode dans certaines contrées). Si c'est un Windows, il y a des chances qu'il opte pour un langage .net. Cela dit, tout est possible, il y en a même qui font du Javascript dans nodes.js dans Azure.

Côté client, le développement d'une application web métier moderne se fait nécessairement, à mon sens, par une application SPA (Single Page Application), c.à.d. une application qui au lieu de recevoir le HTML du serveur, le produit et le modifie grâce au Javascript en local dans le navigateur.

Concernant les AppStores, si on ne tient pas compte des outils de Xamarin, sur les plateformes mobile iOS le choix du langage est limité en gros à Objective C et Javascript. Si vous faites le choix d'un développement natif, il ne restera que l'Objective C. Sous Android le choix est plus ouvert : il y a le C, le C++, Java et quelques autres langages qui tournent sous le JVM comme Scala ou Clojure, et bien sûr Javascript. Sous Windows 8, le Javascript est un langage de programmation natif au même titre que le C++ ou le C#. Grâce à des outils comme PhoneGap, beaucoup d'applications métier peuvent être développées sur ces plateformes en utilisant les technologies du web. Le Javascript est donc utilisable avec l'avantage majeur de permettre une réutilisation importante entre les versions iOS, Android et Windows 8. Les applications sur le Chrome Web Store ou Firefox Marketplace sont écrites en Javascript.

Le format d'échanges de données à travers les API REST (à la mode) est de plus en plus le JSON, un sous-ensemble du Javascript. Bref vous l'aurez compris, un développeur moderne ne peut pas faire l'impasse du Javascript aujourd'hui.

Enfin il y a le poids de l'histoire. Au même titre que le FORTRAN et le COBOL sont indétrônables dans leur niches, vous n'introduirez pas du .net là où il y a un existant Java important, ni du Java là où il y a un existant important en C++, puisque vous n'avez pas envie de perdre du temps à créer des incompatibilités et des bugs entre les chaines de caractères, les dates et les tableaux en passant d'un environnement à l'autre.

L'informatique moderne est riche, multiplateforme, distribuée, de plus en plus dans les nuages. Elle n'en est qu'à ses débuts, tout reste à faire. Depuis 50 ans, les langages de programmation n'offrent pour la plupart que des syntaxes différentes pour écrire des « if » et des « while » avec ou sans accolades. Les problèmes présents et futurs de l'informatique ne se résoudront pas à travers un langage nouveau. Choisissez-en quelques-uns, maîtrisez leurs bibliothèques et environnements. Mes choix à moi sont le C#, le Javascript et si je devais choisir un nouveau langage à apprendre, ce serait certainement le F#.

Codez bien !

Frédéric **Fadel**
Cofondateur d'Aspectize : de l'idée au produit, une approche lean pour le développement des application métier SaaS et Mobile. Cofondateur de Winwise, Frédéric programme depuis 30 ans.

Choisir son langage de programmation

Quel langage de programmation devrais-je choisir ? Beaucoup d'articles sur Internet apportent des réponses, ou des éléments de réponse plus ou moins détaillés à cette question. Il n'y a que vous qui pouvez y répondre, en fonction des objectifs que vous visez. C'est pourquoi cet article s'intéressera à détailler les principes des différentes familles de programmation, ainsi que les domaines métiers dans lesquels ils sont utilisés de nos jours. Commençons par le commencement : d'où viennent les langages de programmation utilisés aujourd'hui ?

Tout débute dans les années 1940 avec le langage machine. Le langage machine est un langage dit "bas niveau". Il est composé d'une suite d'instructions directement exécutables par le processeur (CPU). Ces instructions sont des suites de bits dont la taille peut-être fixe ou variable en fonction de l'architecture du processeur.

```
<code label="Extrait de code assembleur>
…
CALL_CALC:
          MOV   AL,INT_NUMBER
          MOV   BYTE PTR INT_CODE,AL
          DB    0CDh       ; INT
INT_CODE:
          DB    00h
          NOP
          NOP

CALC_EXIT:
          INT   20h
...
</code>
```

Ces instructions qui composent le programme sont intimement liées à l'architecture externe du processeur. Parmi les architectures existantes, il est possible de citer le CISC - incluant la famille de processeurs x86 - et le RISC comprenant notamment les processeurs PowerPC et SPARC.
L'écriture de programmes en langage machine est difficile car source d'erreur, c'est pourquoi dès le début les langages assembleurs ont été créés. Catégorisés comme langages de seconde génération, Ils ont introduit des symboles dits "mnémoniques" qui représentent les instructions binaires du langage machine. Ces symboles, plus lisibles et facilement mémorisables, ont permis de simplifier l'écriture des programmes.
Dès les années 1950, les premiers langages "haut niveau" sont créés. Un langage est dit haut niveau lorsque sa notation est proche des langages naturels. Un des premiers langages haut niveau est le Fortran. Il s'agit d'un langage dédié aux calculs mathématiques qui, dans sa deuxième version publiée en 1958, introduit pour la première fois les fonctions, les boucles et une première structure de contrôle "FOR" primitif. Il s'agit du premier langage de 3ème génération.
A la fin des années 1950 et pendant toute la période des années 1960, une multitude de langages spécialisés sont conçus (Lisp, COBOL, Simula etc.), ayant pour objectifs de traiter des problèmes relatifs à des domaines spécialisés tel que les mathématiques, l'intelligence artificielle ou encore les applications de gestion pour COBOL. A la même époque un comité réunissant des informaticiens Américains et Européens publie le langage ALGOL 58, conçu pour être plus général, il apportera de nombreuses innovations :

- Les structures de code en blocs imbriqués : cela permet au sein d'une même procédure d'isoler des blocs de codes les uns des autres.
- La notion de scope des variables : il est désormais possible de créer des variables dont la portée se limite à un bloc ou une procédure.

Avec ces innovations, les langages ALGOL (ALGOL 58 et ses implémentations suivantes) influenceront grandement beaucoup de langages actuels, notamment C, C++ et Python.
Entre la fin des années 1960 (qui marquent les débuts de la micro-informatique) et le début des années 1980, la majorité des paradigmes de programmation utilisés de nos jours sont conçus :

- Les langages C et Pascal représentent les langages impératifs
- Smalltalk et Simula : la programmation orientée objets
- Prolog : la programmation logique
- ML et Scheme : la programmation fonctionnelle.

Ces langages ont inspiré la majorité des langages actuels et certains sont même toujours utilisés dans le développement de logiciels commerciaux. C'est aussi dans les années 1970 que le langage SQL est inventé, conçu initialement pour la gestion des données au sein de systèmes de gestion de bases de données relationnelles.
Pendant les années 1980, l'enjeu n'est plus la création de nouveau paradigmes, mais leur consolidation. C'est à cette époque que le C++ a été conçu, réunissant les capacités du langage C en terme de programmation système et les concepts de la programmation objet. Les années 1980 sont aussi une époque d'innovation dans la conception des programmes avec l'émergence de la notion de modules : des unités de programmes paramétrables qui peuvent être largement réutilisés. Il s'agit de la première forme de programmation générique, introduite notamment avec le langage Ada.
Les années 1990 sont marquées par la généralisation de la programmation orientation objet grâce aux performances des micro-ordinateurs, mais aussi par l'avènement d'Internet. Les langages Python, Ruby, Java, JavaScript et PHP très largement utilisés aujourd'hui sont alors créés.
A cette époque une forte volonté d''amélioration de la productivité des développeurs entraîne la création de langages permettant un "développement rapide d'application" (RAD : *rapid application development*). Des langages comme Pascal Object, Visual Basic et Java sont créés dans ce but. Ces langages viennent avec des outils qui font gagner du temps aux développeurs :
Les IDE (integrated development environment) : environnements de développement apportant des fonctionnalités comme l'autocomplétion du code, l'identation automatique et la coloration syntaxique.
Les Garbage Collector ("ramasse-miettes" en Français) : programme autonome responsable du recyclage de la mémoire inutilisée.
Désormais, la gestion de la mémoire n'est plus entièrement à la charge du développeur **Fig.1**.
Les années 2000 s'inscrivent dans la continuité des années 1990 : le web devient un enjeu de plus en plus important pour les entreprises et

les particuliers. Les améliorations techniques, notamment au niveau réseau, permettent la réalisation d'applications de plus en plus riches. Les langages créés depuis les années 2000 n'apportent pas de changements fondamentaux en termes de conception par rapport à leurs prédécesseurs. En revanche, l'objectif de ces langages est souvent de rassembler les avantages des différents langages existants. En effet, les langages ont une forte influence les uns sur les autres. Cela ne concerne pas seulement les nouveaux langages qui s'inspirent des anciens. Les nouvelles versions de langages existants vont parfois intégrer des fonctionnalités mises en oeuvre dans des langages plus récents Fig.2.

COMMENT CHOISIR ?

Nous l'avons vu, depuis les premiers langages de programmation créés dans les années 1940, de nombreuses innovations conceptuelles ont fait émerger des familles de langages utilisant des syntaxes et des paradigmes de programmation différents. Le choix d'un langage peut dépendre de nombreux facteurs, c'est pourquoi nous étudierons les différents critères de sélection.

Les paradigmes de programmation

Il est important de prendre en compte le choix du (ou des) paradigme(s) de programmation qu'un langage peut vous offrir. Un paradigme de programmation est un style fondamental qui influe sur la façon dont vous allez concevoir votre application pour répondre à une problématique. Il impacte directement la structure du programme et la façon dont ses éléments sont organisés.

De par leurs différences au niveau conceptuel, tous les paradigmes ne sont pas utilisables dans tous les langages. Par exemple, le langage C n'a pas été conçu pour nativement supporter la programmation orientée objet. Pourtant, rien n'empêchait ses créateurs, Dennis Ritchie and Ken Thompson, d'intégrer cette capacité au langage. En effet, le concept de programmation orientée objet est né dans les années 1960 en Norvège, et le premier langage à supporter ce paradigme est apparu au milieu des années 1960 : Simula. Rappelons-le, le langage C fut développé entre 1969 et 1973, donc plusieurs années après le premier langage orienté objet. Si le langage C n'intègre aucun concept de la programmation orientée objet, c'est avant tout parce que ce langage fut conçu dans le but de réécrire le système Unix, à l'époque en assembleur, avec un langage haut-niveau. Les langages existants à l'époque comme le B ne leur convenaient pas, car ne permettaient pas de reprendre toutes les fonctionnalités développées en langage assembleur. En s'inspirant de langages impératifs tels que B ou ALGOL 68, aucune notion de programmation objet ne fut implantée dans le langage C. Ce n'est qu'en 1983 que la programmation orientée objet rencontre le langage C à travers la création du langage C++, intégrant nativement les concepts de la programmation orientée objet au langage C.

Quels sont les paradigmes de programmation principalement utilisés aujourd'hui ?

Parmi les nombreux paradigmes de programmation, il est possible de distinguer trois grandes familles de paradigmes très largement utilisées :

- La programmation impérative,
- La programmation déclarative,
- La programmation orientée objet.

La programmation impérative

Il s'agit historiquement du premier paradigme conçu. Ce paradigme permet de décrire les traitements à effectuer par la machine sous forme de séquences d'instructions à exécuter pour modifier l'état du

Fig.1

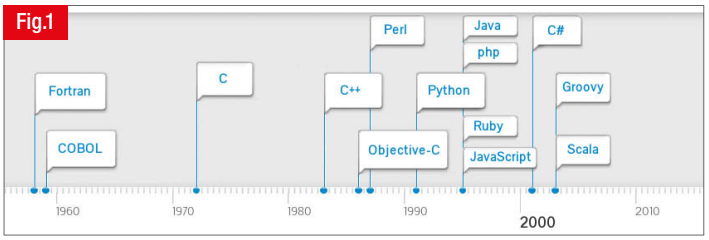
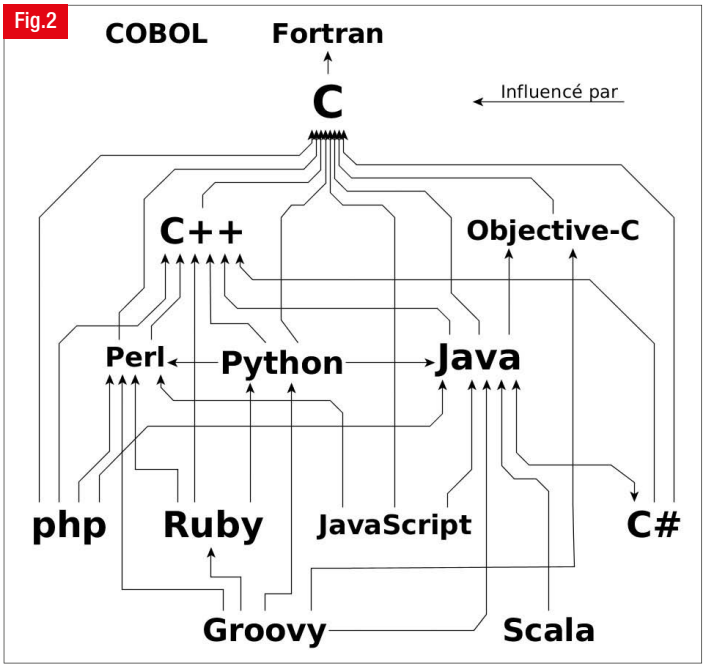


Fig.2



programme. L'intégralité (ou presque) des processeurs sont conçus sur ce principe. En effet, les langages assembleurs et même le langage machine exécuté par le processeur décrivent sous formes d'instructions séquentielles les opérations à effectuer. Il s'agit du paradigme le plus répandu parmi les langages de programmation : Fortran, C, C++, Python, PHP ou encore Java sont quelques exemple de langages largement utilisés qui permettent ce type de programmation.

Exemple de programme en JavaScript pour multiplier par 2 tous les nombres d'un tableau d'une façon impérative :

```
var numbers = [1,2,3,4]
var doubled = []

for(var i = 0; i < numbers.length; i++) {
  var newNumber = numbers[i] * 2
  doubled.push(newNumber)
}
console.log(doubled) // affiche : [2,4,6,8]
```

Exemple de code impératif en C# pour ajouter 1 à tous les nombres d'une liste puis additionner le tout :

```
int sum = 0;
foreach (int i in mylist)
{
    sum += (i + 1);
}
```

Les principales caractéristiques de la programmation impérative sont :

- L'affectation directe de valeurs à des variables,
- La notion de variable globale, utilisable partout dans le programme.

La programmation déclarative

La programmation déclarative est souvent définie comme “n’étant pas” impérative. Si la programmation impérative décrit “comment” le programme doit se comporter sous la forme de séquence d’instructions modifiant son état, la programmation déclarative, elle, consiste à décrire le résultat attendu et laisse la machine se débrouiller pour le “comment”. Parmi les langages permettant (et pour certains, imposant) l’utilisation de ce paradigme, quelques exemples sont : SQL, HTML, Prolog, Clojure et Scheme.
Si l’on reprend l’exemple du programme JavaScript précédent, il est possible de le réécrire avec une approche déclarative (et même fonctionnelle) :

```
var numbers = [1,2,3,4]

var doubled = numbers.map(function(n) {
  return n * 2
})
console.log(doubled) // affiche : [2,4,6,8]
```

Dans cet exemple, nous utilisons la fonction “Array.map” qui créé un nouveau tableau à partir du tableau existant “numbers”, pour lequel chaque élément est créé en faisant passer les éléments du tableau original “numbers” dans la fonction transmise en argument à la fonction map (function (n) {return n * 2}). Ainsi, dans cette exemple la fonction map nous soustrait de faire l’itération (le “comment”) et nous permet de nous concentrer sur la définition mathématique du résultat attendu.
Idem pour l’exemple en C#, réécrit de manière déclarative :

```
int result = mylist.Select(i => i + 1).Sum();
```

Au sein de la famille “déclarative”, il est possible d’identifier plusieurs sous-ensembles principaux :

- La programmation fonctionnelle : un style de programmation consistant à considérer les calculs en tant qu'évaluation de fonctions mathématiques n’introduisant pas de changement d'état ni de modification des données du programme. Un langage est dit fonctionnel lorsque sa syntaxe et ses caractéristiques encouragent la programmation fonctionnelle, par exemple Haskell, Scheme, Clojure ou Scala.
- La programmation logique : une forme de programmation qui permet de définir une application à l’aide d’un ensemble de faits élémentaires et de règles logiques. Ces faits et ces règles sont ensuite exploités par un démonstrateur de théorème ou moteur d'inférence, en réaction à une question ou requête. Le langage Prolog est un exemple d’application de ce style.

Exemple en prolog :
On définit les faits suivants dans un fichier .pl :

```
animal(chien).
animal(chat).
prenom(paul).
prenom(pierre).
prenom(jean).
```

Ce programme affirme les faits suivants :

- chien et chat sont des animaux
- paul, pierre et jean sont des prénoms

Lorsque l’on ouvre ce fichier de faits dans l’interpréteur Prolog, on peut ensuite interroger le programme :

```
?- prenom(jean).
```

L’interpréteur retourne “true” indiquant que “jean” est effectivement un prénom.
Les caractéristiques de la programmation fonctionnelle sont :

- La possibilité d’exprimer le lambda-calcul : langage de programmation théorique dans lequel tout élément est une fonction, exprimée sous la forme d’une expression qui peut contenir des fonctions qui ne sont pas encore définies et qui sont alors remplacées par des variables. Ex : <code OCaml>(fun x -> x+3) 10</code>. Cette expression définit une fonction qui pour tout argument x renvoie x+3. Ici, l’argument “10” est donné à la fonction. Le résultat de cette expression est donc “13”.
- La récursion
- La limitation complète des effets de bord : un effet de bord caractérise le changement d’état d’un élément (une variable par exemple) indépendant des arguments fournis en entrée de la fonction. En limitant au maximum les effets de bord, cela permet de prédire au mieux le comportement du programme quels que soient ses entrants.

Les caractéristiques de la programmation logique sont :

- La logique : le langage est un sous-ensemble de la logique et une exécution est appelée “preuve”.
- La symbolique : les données manipulées sont principalement des symboles.
- Déclaratif : le “quoi faire” plutôt que “comment faire”
- Relationnel : un programme logique décrit un eétat du monde en termes de données (les faits) et de prédicats (relations ou règles) entre ces données.
- Indéterminisme : le résultat est l’ensemble des données qui vérifient une question dans une description donnée du monde.
- Haut niveau : aucune gestion de la mémoire et abstraction du caractère impératif de la machine.

La programmation orientéé objet

Ce style consiste à créer des structures appelées “objets” qui possèdent des données (appelées attributs) et des procédures (appelées méthodes). Cette conception a pour objectif de permettre la représentation, via le langage informatique, d’un concept ou de n’importe quelle entité du monde physique (une voiture par exemple). Les attributs sont des données qui décrivent les caractéristiques de l’objet tandis que les méthodes sont des procédures qui décrivent son comportement. Le but de la programmation orientée objet est donc de manipuler ces objets et de les faire interagir entre eux afin de répondre à une problématique.
Par exemple, dans le cadre de la réalisation d’un programme permettant de gérer un système de location de voiture, une conception objet de ce système pourrait consister à définir un objet “Voiture” avec pour attributs une couleur et une marque, ainsi qu’un objet “Personne” avec pour attribut un nom, un prénom et un âge, puis à les faire interagir au travers de leurs méthodes respectives ou par le biais de méthodes d’autres objets.

```
package fr.code.sample;

public class Voiture {
 private String couleur;

 private String marque;

 public String getCouleur() {
       return couleur;
```

```
    }

    public void setCouleur(String couleur) {
        this.couleur = couleur;
    }

    public String getMarque() {
          return marque;
    }

    public void setMarque(String marque) {
        this.marque = marque;
    }
}
```

La programmation orientée objet est caractérisée par plusieurs grands concepts :

- Les objets
- L'héritage : caractérise le fait qu'un objet soit basé sur un autre. On parle alors de hiérarchie entre ces objets nommés "parents" et "enfants". Il s'agit d'un mécanisme permettant de réutiliser du code défini dans le "parent" au sein de "l'enfant" sans le déclarer à nouveau.
- La possibilité de masquer l'information : les objets sont définis par des attributs qui contiennent des données et des méthodes qui décrivent des comportements de l'objet. En programmation orientée objet, le programme est défini par ses objets et les interactions entre ces objets. Dans le but de minimiser les dépendances entre les implémentations concrètes des différents objets, et donc l'impact des modifications d'un objet sur un autre, il faut que les objets se voient entre eux comme des boîtes noires. Leur comportement est connu mais pas leur implémentation.

L'encapsulation est une des techniques permettant d'appliquer ce concept : on utilise un niveau de visibilité des attributs restreint à la classe (private) et des méthodes accessibles de l'extérieur qui garantissent le contrôle des accès aux valeurs des variables.
L'utilisation d'interfaces permet de masquer l'implémentation des méthodes ainsi que des structures de données d'un objet aux autres, appelés "clients". Cela permet par exemple de changer l'implémentation d'une méthode décrite dans une interface sans avoir à modifier le code des autres objets qui l'utilisent.

- Le polymorphisme : le principe est de définir un code dont le comportement varie en fonction du type d'objet qui l'utilise. On distingue trois formes de polymorphisme :
  - Le polymorphisme "Ad hoc" : il est possible de créer deux méthodes dans un objet dont le nom est identique, le nombre d'arguments aussi, mais le type des arguments diffèrent.

Ex :

```
function Add( x, y : Integer ) : Integer;
begin
    Add := x + y
end;

function Add( s, t : String ) : String;
begin
    Add := Concat( s, t )
end;

begin
    Writeln(Add(1, 2));
    Writeln(Add('Hello, ', 'World!'));
end.
```

  - De cette manière, la méthode "Add" qui sera utilisée dépend du type de ses paramètres.
- Le polymorphisme paramétrique : souvent appelé "generics" ou "templates" dans les langages de programmation, cette fonctionnalité permet de rendre les méthodes et attributs génériques. Cela permet de définir du code indépendant du type de l'objet qu'il est amené à manipuler.

Exemple :

```
public <T> T getAdapter(Class<T> itf) {
      T facet = (T) getAdapters().get(itf);
      if (facet == null) {
            facet = findAdapter(itf);
            if (facet != null) {
                  adapters.put(itf, facet);
            }
      }
      return facet;
}
```

Ce code garantit que n'importe quel type de "Class" peut être fourni en paramètre de la fonction getAdapter et que l'objet retourné est bien une instance de type T (représentant n'importe quel type d'objet) de la classe "Class<T>" passée en paramètre.

- Le polymorphisme par sous-typage (ou polymorphisme d'inclusion) : le principe est qu'une méthode acceptant un paramètre d'un type "A" acceptera aussi un paramètre dont le type "B" est un sous-type de "A". Un type d'objet peut en effet être étendu en sous-types. Par exemple, on peut considérer "chat" et "chien" comme des sous-type de "animal" :

```
abstract class Animal {
    abstract String talk();
}

class Cat extends Animal {
    String talk() { return "Meow!"; }
}

class Dog extends Animal {
    String talk() { return "Woof!"; }
}

void lets_hear(Animal a) {
    println(a.talk());
}

void main() {
    lets_hear(new Cat());
    lets_hear(new Dog());
}
```

Dans cet exemple, on a une procédure "lets_hear" acceptant un paramètre de type "Animal" qui peut être appelé avec un objet de type "Chat" ou "Chien" à la place, puisqu'il s'agit de sous-types du type "Animal".

Langages et paradigmes

- La programmation impérative (C, C++, C#, Java, PHP, Python...)
- La programmation déclarative (SQL, CSS...)
- La programmation fonctionnelle (Erlang, Haskell, Lisp, Clojure, Scala, C#, Java...)
- La programmation logique (Prolog, Datalog...)
- La programmation orientée objet (C++, C#, Objective-C, Java, PHP, Python, Ruby, Scala)

Structure et syntaxe

Un autre critère qui fait que l'on préfère un langage par rapport à un autre est sa structure et sa syntaxe. Comme évoqué précédemment, les paradigmes de programmation ont une influence certaine sur la structure.

Types de données

Il s'agit d'un des points de différenciation importants entre les langages : le typage des données. Certains langages sont plus ou moins stricts ou plus ou moins permissifs que d'autres sur ce point. On parle de typage statique quand la majorité des vérifications de type sont effectuées au moment de la compilation du programme. La compilation garantit ainsi qu'aucune erreur de type ne se produira à l'exécution. A l'inverse, on parle de typage dynamique lorsque ces vérifications ont lieu à l'exécution du programme.
Exemple de code à typage statique :

```
int i = 42;
char y[] = "37";
```

Dans l'exemple ci-dessus, le type des variables "i" et "y" est déclaré directement. Le compilateur est alors à même de vérifier que les assignations sont correctes en termes de type.
Exemple de code à typage dynamique :

```
var x = 5
var y = "37"
var z = x + y
```

Ici le mot clé "var" se substitue à l'utilisation d'un type comme dans l'exemple précédent. Lors de l'évaluation des expressions ci-dessus au runtime, il apparaitra que la variable x est de type "number", la variable y de type "string" et la variable z de type.... "string" ! En effet, l'opérateur "+" ayant à la fois un "number" et un "string" en opérande se comportera comme l'opérateur de concaténation, et la valeur de x se trouvera implicitement convertie en string. La valeur de z est ainsi "537". En revanche, si l'on déclare <code javascript>var y = 37</var>, lors de l'exécution z sera résolu comme étant de type "number", et sa valeur sera alors le résultat de l'addition : 42.
D'une manière générale l'utilisation d'un typage statique permet au compilateur d'effectuer des optimisations du code machine généré et donc de meilleures performances comparées aux langages à typage dynamique. Certains langages à typage dynamique permettent optionnellement la déclaration de type pour cette raison (Common Lisp par exemple).
Le typage dynamique a aussi ses avantages : il permet un temps de compilation plus rapide et permet, dans le cas des langages interprétés, de charger dynamiquement du nouveau code. L'autre avantage qui en découle est une réduction du temps d'attente dans le cycle de développement (modification du code / compilation / test / debug).
Certains développeurs préfèrent les langages à typage statique parce qu'ils alertent le développeur dès la compilation lorsqu'il y a des erreurs de types. D'autres préfèrent les langages à typage dynamique clamant qu'ils permettent de développer plus rapidement et que les erreurs de typage ne sont qu'une toute petite partie des erreurs que l'on rencontre dans un programme.

	Typage statique	Typage dynamique
Langages	C, C++, C#, COBOL, Fortran, Haskell, Java, Scala	Groovy, JavaScript, Lisp, Objective-C, PHP, Ruby, Python

En plus des notions de typage statique et dynamique, on parle aussi de "sûreté" du typage. Un langage est considéré "type-safe" s'il ne permet pas d'opération ou de conversion qui violerait les règles du système de typage. Il est considéré comme "memory-safe" s'il ne permet pas l'accès à des emplacements mémoire qui ne lui sont pas assignés. Par exemple, un langage memory-safe verifiera les bordures d'un tableau ou préviendra à la compilation (ou à l'exécution) qu'un accès est fait en dehors des limites du tableau. C'est le cas du langage Java par exemple. A l'inverse, un langage qui n'est pas memory-safe autorisera l'accès à des emplacements mémoires non assignés, et donc la récupération de données inattendues, comme dans le cas du langage C. L'utilisation d'un langage non type-safe ou memory-safe est plus risqué pour le développeur qui peut faire des erreurs dans la manipulation de tableaux sans rencontrer d'erreur dans l'exécution de son programme. Il est donc recommandé pour un débutant de commencer avec un langage type-safe et memory-safe car ce dernier fera les vérifications nécessaires à la compilation ou lancera une erreur explicite à l'exécution, permettant une rapide correction de l'anomalie.

La syntaxe

Les langages mettent à disposition du développeur un ensemble de mot clés réservés, de caractères spéciaux et d'opérateurs qui permettent l'expression de ce langage et la structuration du contenu du programme. Parmi les différents langages évoqués jusqu'à présent, voici un ensemble d'exemples de codes illustrant la syntaxe utilisée et la

Langage	Typage statique / dynamique	Sûreté
Assembleur	aucun	non-sûr
C	statique	non-sûr
C++	statique	non-sûr
C#	statique	les deux
COBOL	statique	non-sûr
Fortran	statique	sûr
Groovy	dynamique	sûr
Haskell	statique	sûr
Java	statique	sûr
JavaScript	dynamique	sûr
Lisp	dynamique	sûr
Objective-C	dynamique	non-sûr
PHP	dynamique	sûr
Python	dynamique	sûr
Ruby	dynamique	sûr
Scala	statique	sûr

structure de la programmation dans différents langages :

```
<code assembleur x86 (DOS, FASM)>
; FASM example of writing 16-bit DOS .COM program
; Compile: "FASM HELLO.ASM HELLO.COM"
  org  $100
  use16
  mov  ah,9
```

```
  mov  dx,xhello
  int  $21    ; DOS call: text output
  mov  ah,$4C
  int  $21    ; Return to DOS
xhello db 'Hello world !!!$'
</code>
```

```
<code C>
#include <stdio.h>

 int main(void)
 {
    printf("Hello, world!\n");
    return 0;
 }
</code>
```

```
<code C++>
#include <iostream>

using namespace std;

int main()
{
    cout << "Hello, World!" << endl;
    return 0;
}
</code>
```

```
<code C#>
using System;

internal static class HelloWorld
{
    private static void Main()
    {
        Console.WriteLine("Hello, world!");
    }
}
</code>
```

```
<code C# (une seule ligne)>
class s{static void Main(){System.Console.WriteLine("Hello, world!");}}
</code>
<code COBOL (version condensée)>
IDENTIFICATION DIVISION.
PROGRAM-ID.  HELLO-WORLD.
PROCEDURE DIVISION.
    DISPLAY "Hello, world!"
    STOP RUN.
</code>
```

```
<code Groovy>
println "Hello, world!"
</code>
```

```
<code Haskell>
main = putStrLn "Hello, world!"
</code>
<code javascript>
alert('Hello, world!');
</code>
```

```
<code java>
/* HelloWorld.java
 */

public class HelloWorld
{
    public static void main(String[] args) {
    System.out.println("Hello World!");
    }
}
</code>
```

```
<code Common Lisp>
(write-line "Hello, world!")
</code>
```

```
<code objective-C>
#import <stdio.h>

int main (int argc, const char *argv[])
{
    printf ("Hello, world!\n");
    return 0;
}
</code>
```

```
<code PHP>
<? echo "Hello, world!"?>
</code>
<code Python>
import sys
sys.stdout.write("Hello, world!\n")
</code>
```

```
<code Ruby>
class String
     def say
         puts self
     end
 end
 'Hello, world!'.say
</code>
<code Scala>
object HelloWorld extends App {
  println("Hello, world!")
}
</code>
```

Au travers de ces exemples, on constate des ressemblances mais aussi beaucoup de différences. L'exemple choisi ne met pas en valeur tous les éléments de syntaxe des différents langages. Il permet toutefois de comparer facilement comment les différents langages présentés permettent de répondre à la même problématique : afficher "Hello, world !".

Compil VS interprété

L'implémentation (réalisation concrète) d'un langage de programmation peut être interprétée ou compilée. Un langage de programmation peut

avoir une implémentation compilée, et une autre, interprétée. La compilation désigne le processus de transformation d'un code source d'un langage à un autre. Cette technique est principalement utilisée dans le but de traduire un code source vers un langage bas niveau (assembleur ou code machine) directement exécutable par la machine. On parle de langage interprété lorsqu'aucun mécanisme de traduction du code source n'est nécessaire pour l'exécution du programme par un interpréteur. L'interpréteur est le programme responsable de l'exécution du code source non traduit. Contrairement à un compilateur, il ne produit pas de code exécutable sauvegardé dans un fichier. La transformation du code source en code machine se fait à la volée et est exécuté ligne après ligne.

Une analogie simple pour mieux comprendre : vous devez donner des ordres à une personne qui ne parle que chinois, or la seule langue que vous connaissez est le français. Deux approches sont possibles si l'on considère que la liste de vos ordres est déjà définie :

- Soit vous faite traduire en chinois à l'avance et présentez la liste d'ordre écrite en chinois (compilé)
- Soit vous faites appel à un interprète qui, pendant que vous lui énoncez les ordres en français, se charge de donner les ordres en chinois, à l'oral (interprété).

La personne qui parle chinois (et qui représente la machine dans notre exemple) effectuera les ordres donnés en chinois, mais il y a plusieurs processus pour donner les instructions.

Avantages et inconvénients des deux techniques

Parce qu'il est déjà écrit en langage machine, un programme compilé est plus rapidement exécuté par la machine qu'un code source interprété. En revanche, comme nous l'avons évoqué précédemment, le langage machine n'est pas universel. Par conséquent, un programme compilé en langage machine n'est directement exécutable que sur un ensemble de plateformes et systèmes d'exploitations cibles de la compilation. Un avantage des programmes en langage interprété est justement qu'ils ne sont pas spécifiques à la plateforme d'exécution cible du langage. Ils sont donc beaucoup plus portables, la seul contrainte étant qu'un logiciel interpréteur du langage soit présent sur le système. Un des points faibles des langages interprétés, autre que les performances, en est aussi une de leurs forces : le code source est accessible directement. Contrairement à un programme compilé pour lequel il n'est pas possible de modifier le code, la modification du code source d'un programme en PHP (par exemple) permet un gain de temps certain pour la correction de bug par exemple. Cela pose aussi un problème concernant l'intégrité du programme. Puisque le code source est accessible, modifiable et réinterprété directement, cela peut poser des problèmes de sécurité. Un exemple typique ; le HTML et le JavaScript dans les navigateurs. Il est tout à fait possible pour l'utilisateur du navigateur de modifier le code JavaScript inclus dans la page HTML qu'il reçoit, ou même de bloquer l'interprétation du JavaScript. Ces problématiques ne peuvent être entièrement évitées avec les langages interprétés.

A mi-chemin

Les programmes compilés en langage machine sont plus rapides mais exécutables sur un nombre restreint de plateformes. Les programmes en langage interprété, eux, sont théoriquement portables sur toutes les plateformes, mais sont moins performants à l'exécution et peuvent présenter des risques de sécurité car leur intégrité peut-être mise à mal.

A mi-chemin entre la compilation et l'interprétation : la machine virtuelle. Théorisée dans les années 1930, une machine virtuelle est une illusion d'un appareil informatique créée par un logiciel d'émulation. Le logiciel simule la présence de ressources matérielles et logicielles telles que la mémoire, le processeur, le disque dur, voire le système d'exploitation et les pilotes, permettant d'exécuter des programmes dans les mêmes conditions que celles de la machine simulée. C'est seulement dans les années 1970 que les premières implémentations de machine virtuelle voient le jour. A la fin des années 1990, Sun Microsystem conçoit la première machine virtuelle destinée à l'exécution de programmes en Java. Le slogan "write once, run anywhere" annonce les objectifs de la Java Virtual Machine : Java devient un langage cross-platform.

En effet, la JVM est notamment composée d'un interpréteur de bytecode Java, un langage intermédiaire plus bas niveau que Java. Ce bytecode est donc indépendant des plateformes d'exécution, mais nécessite une JVM pour être exécuté. La transformation du code Java en bytecode est un simple processus de compilation, sauf que la plateforme cible est la JVM. Parce que le langage Java est d'abord compilé en bytecode, puis interprété par une JVM, on dit que Java est un langage semi-compilé.

Aux débuts de la JVM, Java souffrait de problèmes de performance à l'exécution, comme les autres langages interprétés. Une avancée significative en termes de performance fut l'implémentation dans la JVM d'un JIT (Just In Time compiler). Depuis, le bytecode est interprété lors des premiers appels au programme puis compilé en langage machine, garantissant des performances identiques aux langages compilés lors d'un deuxième passage sur les segments de bytecodes déjà exécuté.

La JVM, bien que destinée à l'exécution de programmes Java, joue un rôle important dans l'évolution des langages interprétés d'aujourd'hui. De plus en plus d'implémentation de langages initialement interprétés comme PHP, Ruby ou Python voient le jour sous la forme de langages qui peuvent être compilés en bytecode Java afin de tirer parti des avantages de la JVM. La JVM permet aujourd'hui d'allier les avantages des deux mondes : les performances d'exécution des langages compilés et la portabilité des langages interprétés.

Langages interprétés	Langages compilés	Langages semi-compilés
JavaScript	C	Groovy
Lisp	C++	Java
PHP	C#	Lisp
Python	COBOL	Python
Ruby	Fortran	Scala
	Haskell	
	Lisp	
	Objective-C	

Critère de choix : la difficulté d'apprentissage

Si votre objectif est d'être rapidement opérationnel pour "programmer", une première approche dans le choix de votre langage est de considérer les langages de haut-niveau, c'est à dire à forte abstraction. En effet, la programmation assembleur est beaucoup plus complexe que la programmation JavaScript, par exemple, car aucun mécanisme ne vous facilite les choses et il vous faut connaître et maîtriser beaucoup de concepts liés au fonctionnement détaillé des ordinateurs si vous souhaitez écrire un simple programme. Il vous faut choisir un langage orienté au maximum vers "l'humain" plutôt que vers la machine.

Quels que soient les langages, la structure, la syntaxe et les paradigmes de programmation supportés seront des éléments à apprendre pour programmer. Toujours dans l'objectif d'être opérationnel le plus rapidement possible pour faire des programmes basiques, il vous faut vous orienter vers les langages avec les caractéristiques suivantes :

- Un langage interprété : pas besoin de connaître le processus de compilation puisqu'il n'y en a pas, portable, processus de développement avec itération rapide : dès la modification du code on peut voir le résultat.
- Typage : préférez un langage avec typage statique si vous souhaitez

être attentif aux éléments que vous manipulez, mais préférez un typage dynamique dans le cas ou vous souhaitez plus de libertés quant à la manipulation des variables (Attention, le debug peut s'avérer compliqué).

- Syntaxe simple : moins il y a de mots réservés dans le langage, moins il vous faut les retenir, et plus il est simple d'écrire le programme. A noter que, même si ce n'est pas toujours le cas, plus le nombre de mots clés est faible, plus le langage vous limitera en termes de fonctionnalités.
- Structure : un langage très structuré vous imposera beaucoup de contraintes dans la rédaction de votre programme, mais vous garantira une structure compréhensible à la lecture. A l'inverse, moins le langage apporte d'éléments de structuration du code, plus il est facile de développer (au moins dans un premier temps).
- Type-safe et memory-safe : pour débuter, il est important d'utiliser un langage "sûr", pour lequel l'interpréteur (ou le compileur) saura vous alerter des dangers encourus en manipulant des emplacements mémoire inattendus.

D'une manière générale, les langages qui peuvent être utilisée en tant que script (JavaScript, PHP, Python, Groovy..) sont de bons langages à apprendre si vous souhaitez rapidement monter en compétence et faire vos premiers scripts et programmes. En revanche, si votre objectif est justement de comprendre les principes sous-jacents aux langages énoncés avant, comme la gestion manuelle de la mémoire par exemple, l'apprentissage du langage C ou du C++ vous apportera un niveau de détail très fin dessus. Au niveau intermédiaire, entre le C et les langages interprétés de scripting, les langages Java, C# et Objective-C. Tout en nécessitant une certaine maîtrise de nombreux concepts, la gestion de la mémoire est grandement simplifiée avec l'existence d'outils automatiques d'allocation / récupération de la mémoire.

Le monde du travail

Le choix d'un langage ne se fait pas seulement parce qu'on apprécie sa syntaxe, sa structure et ses styles de programmation. Aujourd'hui, une minorité de langages est utilisée au niveau des entreprises pour la programmation de leurs logiciels. Les langages généralistes et de haut-niveau se sont solidement ancrés dans la plupart des domaines de l'informatique et les domaines d'activité des entreprises. Toutefois, certains secteurs de niche, pour différentes raisons, utilisent des langages utilisés nulle-part ailleurs.

Les domaines d'applications

Le développement web

Avec la constante amélioration technique des réseaux et des ordinateurs, le développement de sites et applications web constitue un marché immense qui atteint la quasi totalité des entreprises. Sites vitrine, sites e-commerce, portails intranets, outils collaboratif ou simples applications de gestion, les applications du développement web sont nombreuses. Le principe de base des architectures logicielles et matérielles en termes de développement web est la notion de client-serveur. Même lorsque l'on parle de Cloud Computing, le principe reste le même : une ou plusieurs machines sont appelées "serveur" et sont responsables d'effectuer les traitements lourds en calcul ou encore les traitements impactant l'ensemble des clients. Le navigateur internet constitue le "client", chargé d'interpréter la réponse du serveur et de communique avec lui.

On distingue alors deux types de programmation :

- La programmation coté serveur :

  Il s'agit d'un nom général donné pour désigner l'ensemble des programmes exécutés coté serveur.

  Objectifs :
  - Traiter les données envoyées par l'utilisateur.
  - Générer et transmettre les pages HTML.
  - Structurer les applications web.
  - Interagir avec les ressources de stockage permanent (SQL, fichiers).

  Exemples de langages utilisés coté serveur :
  - PHP, Java, Ruby, ASP.Net (avec C#), C++, ou même Visual Basic.
- La programmation coté client :

  Expression générale pour désigner l'ensemble des programmes exécutés coté client :

  Objectifs :
  - Réaliser des pages Web interactives et dynamiques
  - Interagir avec des éléments de stockage temporaire (cookies, stockage local)
  - Envoyer des requêtes au serveur et récupérer / traiter les données en réponse.

  Exemples de langages utilisés coté client :
  - JavaScript pour la programmation
  - HTML et CSS pour la description du contenu des pages.

Le développement mobile

Il existe un lien assez étroit entre le développement web et le développement mobile. En effet, beaucoup d'applications mobiles fonctionnent sur la même architecture (client-serveur) et la partie développement "coté serveur" est identique au niveau des langages utilisés. En revanche, les applications "clientes" pour mobile peuvent être développées par différents moyens :

- Soit entièrement avec le langage natif du téléphone :
  - Java pour Android
  - Objective-C pour Apple
  - C# pour Windows Phone
- Soit avec des langages de programmation "coté client" issus du web :
  - De nombreux frameworks et outils permettent de générer des applications multi-plateformes basées sur HTML 5, CSS 3 et Javascript.

Toutefois, bien que l'on retrouve souvent des applications ayant de fortes interactions avec la partie serveur, il est aussi possible de réaliser des applications entièrement standalone, ne nécessitant aucune connexion à Internet pendant l'exécution. Ces applications sont principalement développées en utilisant le langage natif de la plateforme cible.

Le développement pour systèmes embarqués et temps réél

Les systèmes embarqués utilisent généralement des microprocesseurs à basse consommation d'énergie ou des microcontrôleurs, dont la section logicielle est en partie ou entièrement programmée dans le matériel, généralement en mémoire dans une mémoire morte (ROM), FLASH, etc. (on parle alors de *firmware*).

Les contraintes en termes de rapidité d'exécution, de gestion des ressources mémoire sont très fortes dans ce domaine d'application. Cela nécessite l'utilisation de langages suffisamment bas-niveau pour pouvoir optimiser au maximum les programmes. Le langage C est le principal langage utilisé dans ce domaine. Des instructions en langage assembleur peuvent aussi être utilisées dans certains cas.

Le développement pour applications lourdes (desktop)

Le développement d'applications lourdes (par opposition aux applications légères que représentent les applications Web) peut se faire avec de très nombreux langages. La majorité des logiciels très complets et complexes sont développés sur une base de langage C ou C++. Il existe un certain nombre de bibliothèques de fonctions et frameworks qui sont utiles

notamment pour le développement de l'interface graphique du logiciel. On trouve aussi beaucoup d'applications réalisées en Java, surtout pour répondre à la contrainte du "multi-plateforme". Concernant les environnements Microsoft, on trouvera les langages C# et VisualBasic. Enfin, pour ce qui est de la programmation système ou d'applications de calcul lourd, le C et le Python sont principalement utilisés.

Spécificités de certains secteur d'activités

Banque et finance

Dans le secteur de la finance, le langage COBOL est encore majoritairement utilisé pour la gestion des flux financiers et la plupart des applications associées. Bien que des langages comme Java soient tout de même utilisés dans ces secteurs pour des applications non critiques, le COBOL reste majoritaire et cela restera longtemps ainsi. En effet, selon Bill Curtis directeur du CISQ (Consortium for IT Software Quality), les banques vont conserver leurs applications COBOL notamment parce que celles-ci ne présentent pas les mêmes problèmes de sécurité et de performance que Java.

Sciences et recherche

Dans le domaine de la recherche et du calcul scientifique, les langages C, C++ et Fortran sont couramment utilisés mais le cas particulier de ce domaine est l'utilisation du langage (et outil) MATLAB. Ce langage, conçu à la fin des années 1970 sur la base de Fortran notamment, est à la fois, un langage de programmation et un environnement de développement. Il est utilisé à des fins de calculs numériques. MATLAB permet de manipuler des matrices, d'afficher des courbes et des données, de mettre en œuvre des algorithmes, de créer des interfaces utilisateurs, et peut s'interfacer avec d'autres langages comme le C, C++, Java, et Fortran.

Intelligence artificielle

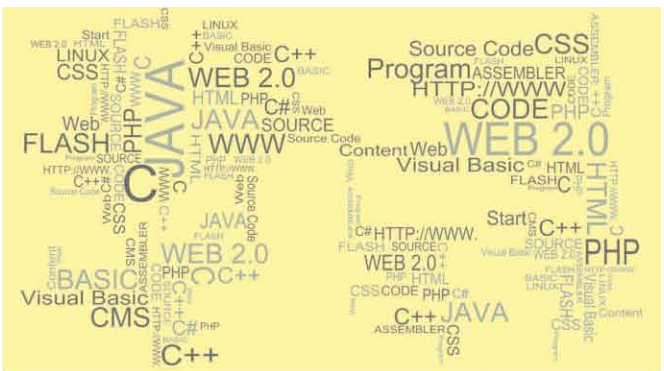
Appliqué à différents domaines comme la banque, le domaine militaire, le jeu vidéo ou encore la médecine, certains langages en plus du C et du C++ sont particulièrement représentés :

- Prolog
- Haskell
- OCaml.

Communautés, frameworks, outils et évolutions

La communauté

Un point à ne pas sous-estimer dans le choix d'un langage de programmation : la communauté. En effet, il faut s'assurer de l'existence d'une communauté vivante et active qui montre la "bonne santé" du langage. Comment trouver la communauté ? Cela s'illustre déjà par l'existence sur de nombreux forum et sites tels que stackoverflow, de questions et réponses récentes et nombreuses. Une autre part importante du travail de la communauté est la communication sous la forme de conférences abordant des aspects précis du langage ou des outils autour. Par exemple, dans le cas du langage Java, la communauté est très active et de nombreuses conférences sont données lors d'évènements plus ou moins majeurs de la programmation, ainsi que lors d'évènements spécifiquement créés pour la promotion de Java (JUG).



Les frameworks et bibliothèques

La grande majorités des langages de programmation utilisés en entreprises ont plus de 20 ans d'existence et ont connu de nombreuses évolutions et changements de versions. De ce fait, de nombreux modules, bilbliothèques de fonctions et frameworks ont vu le jour pour permettre par exemple de réutiliser du code pour répondre à des problématiques récurrentes, ou pour imposer une structure d'application considérée comme efficace pour le développement d'application spécifiques. Ces bibliothèques et frameworks sont de précieux outils pour gagner du temps sans réinventer la roue. Il est donc important de vérifier, avant de choisir un framework ou une bibliothèque, qu'ils sont régulièrement maintenus et qu'il existe une communauté autour. On préférera ainsi ne pas débuter un projet sur la base d'un framework abandonné.

Les outils

Le développement d'un programme comprend de nombreuses phase de travail fastidieuses et sans valeur ajoutée. C'est pourquoi il existe de nombreux outils qui permettent de se concentrer sur les aspects les plus intéressants de la programmation.
Parmi ces outils, le plus utile de tous est sans doute l'IDE (Integrated Developpement Environment) qui permet à la fois de faciliter les étapes de compilation et déploiement, mais aussi l'écriture du code avec la présence d'éditeurs à la coloration syntaxique et aux règles de présentation du code associé à votre langage. L'auto-complétion est aussi l'une des fonctionnalités les plus utiles pour un débutant.
Cet outil augmente considérablement la productivité du développeur, c'est pourquoi il est important, lorsque c'est possible de choisir un langage en vérifiant l'existence d'outils de ce type.

L'évolution du langage

De la même manière que pour les frameworks, un langage peut tomber en désuétude voire se cristalliser sur une version, et ne plus jamais connaître d'évolution. Si vous souhaitez développer avec un langage qui continue de s'améliorer, de s'adapter aux changements techniques et technologiques, il est conseillé de travailler sur un langage "qui a de l'avenir'.

CONCLUSION

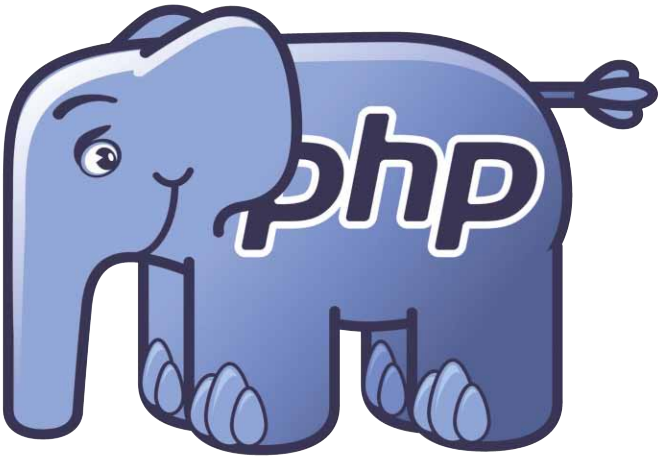
Comme nous l'avons vu, il existe énormément de langages de programmation. Compilés VS interprétés, typage statique VS typage dynamique, programmation fonctionnelle, programmation impérative, programmation orientée objet...
N'oubliez pas que selon vos objectifs personnels ou professionnels, il vous faudra éventuellement choisir un langage adapté au domaine d'activité qui vous intéresse. Enfin, assurez-vous que le langage en question ne tombe pas en désuétude : une communauté vivante, des perspectives d'évolution du langage, et la présence d'outils destinés aux développeurs sont autant de signes qui vous renseigneront sur le caractère "vivant" d'un langage.

Gaël **Nieutin**
Ingénieur étude et développement Java & chef de projet technique - SMILE

Quoi de neuf avec PHP ?

1ere partie

PHP n'a sans doute plus beaucoup de secrets pour vous. Nous ne comptons plus le nombre d'articles publiés sur ce langage dans Programmez !. Pour cette rentrée 2014 et en attendant PHP 7 (il n'y aura pas de version 6), nous voulions faire le point sur PHP et son écosystème : les outils indispensables, PHP 5.6, PHP en mode Cloud, sans oublier les deux frameworks phares, Symfony et Zend. (La rédaction).



Une brève histoire de PHP

Tous les développeurs web connaissent PHP; ils ont dû un jour ou l'autre s'y frotter, que ce soit en formation ou en cours, pour un projet personnel ou professionnel. Le langage possède autant de détracteurs que d'aficionados mais peu savent comment et pourquoi il a été créé à l'origine ni quelles sont les personnes et sociétés qui ont contribué à son évolution technique pour en faire le langage numéro 1 du Web. Voici son histoire.

Le début du commencement

On considère que la création du World Wide Web remonte au début des années 1990, le réseau Internet sur lequel il repose ayant été développé quelques décennies plus tôt. Le WWW a rapidement dû proposer du contenu de plus en plus interactif et arriver à générer des pages dynamiques. PHP, qui est intimement lié à l'expansion fulgurante du Web, est né en 1995. C'est à ce moment que Rasmus Lerdorf, développeur danois, choisit de libérer les sources du compteur de visites qu'il avait créé pour son CV en ligne. A l'origine, le travail de Rasmus consistait en un ensemble de scripts CGI en PERL qu'il a par la suite réécrits en C pour des raisons de performance. La syntaxe restait similaire à celle de PERL, les fonctionnalités étaient assez basiques. C'est pourtant ce qui donne naissance à PHP/FI (Personal Home Page/Form Interpreter). C'est aussi ce qui commence à intéresser les développeurs et aide le système à se répandre. Néanmoins, ce développement organique et empirique donne du grain à moudre aux premiers détracteurs de PHP qui ne reconnaissent pas PHP comme un langage – ce qui n'était effectivement pas l'intention première de Rasmus Lerdorf. Un noyau de développeurs enthousiasmés par le travail de Rasmus permit de sortir une version 2 de PHP/FI en 1997. Mais le manque de cohérence des contributions à PHP/FI a commencé à assombrir l'avenir de cette initiative.

1997, l'année du tournant

En 1997, Zeev Suraski et Andi Gutmans, deux étudiants israéliens, s'intéressent au projet, ne trouvant aucun autre langage répondant à leurs besoins pour leur travail universitaire – le développement d'un shopping cart. Ils commencent donc à améliorer PHP/FI en réécrivant son Parser. PHP change de nom pour l'occasion, perd son FI et prend la forme du fameux acronyme récursif PHP: Hypertext Preprocessor. Après plusieurs mois de tests, PHP 3 est officiellement lancé en 1998. A partir de ce moment, Andi et Zeev travaillent sur la réécriture du cœur de PHP et présentent le Zend Engine, qui reste le cœur de PHP depuis lors.

En mai 2000, PHP 4 et son noyau, le Zend Engine 1.0, sont lancés. Cette version du langage restera comme celle qui a permis l'adhésion à PHP de millions de développeurs dans le monde. Sa longévité extraordinaire en témoigne - le support de PHP 4 par la communauté s'est poursuivi jusqu'en août 2008, et ce malgré la sortie en juillet 2004, de la première version actuelle de PHP 5, embarquant la seconde mouture du Zend Engine offrant un support du paradigme objet bien plus moderne et en phase avec les besoins des projets de plus en plus ambitieux réalisés en PHP. Le support de ce modèle objet par PHP 5.0 a constitué une évolution majeure de PHP, qui se poursuit encore aujourd'hui, chaque nouvelle version du langage poursuivant le support de concepts toujours plus avancés de la programmation objet.

PHP 6, le projet tué dans l'œuf

En 2005, les core développeurs PHP allaient s'attaquer à un gros chantier : le support natif d'Unicode. Le projet fait long feu en raison des changements que ce support nécessite pour le langage. La plupart des fonctionnalités qui devaient être intégrées dans cette version ont été backportées dans des versions de la branche 5 du langage.

Les raisons du succès

Ce qui a rapidement popularisé PHP est sa simplicité d'utilisation. Beaucoup d'autodidactes en programmation ont pu pour la première fois écrire du code et créer leur site web. Pour rejoindre ce nouvel eldorado qu'était le Web en 1997, il fallait savoir programmer. PHP était une porte d'entrée simple pour qui souhaitait créer un site dynamique. Lorsque Yahoo!, à l'époque géant du web, décide d'adopter PHP en 2002. Les regards jusqu'alors dubitatifs reconsidèrent le langage et ses forces : PHP devient un incontournable des projets web.

Quelques années plus tard, l'explosion de Facebook, qui a choisi PHP, est également l'une des raisons qui ont poussé PHP plus encore au sein des entreprises pour des applications de plus en plus conséquentes. Malgré ces succès, nombre de développeurs ont toujours une image négative du langage. Les côtés « quick

and dirty » et « code spaghetti » sont parfois malheureusement toujours associés au langage et ne permettent pas à celui-ci de s'imposer auprès de certains responsables informatiques : pour eux, PHP reste un langage mal structuré, permissif et mal sécurisé, qui n'a pas sa place pour des projets ambitieux et critiques.

Des outils et des hommes

PHP a longtemps été en retard par rapport à d'autres langages tels que .Net et Java si l'on considère le nombre d'outils professionnels disponibles pour aider en développement et à la mise en production. Certains outils ont directement été empruntés et modifiés afin de pouvoir être utilisés avec PHP.

On peut noter également la naissance des différents frameworks génériques, tels que CakePHP, CodeIgniter, Symfony et Zend Framework, qui ont aidé à structurer le développement et participé à son adoption sur des projets de plus en plus conséquents.

Les CMS (Drupal, Wordpress, Joomla...), les plateformes e-commerce (Magento, Prestashop, etc.) ont également un rôle majeur dans l'adoption du langage. Après plusieurs projets de refonte du langage, Zend Technologies, société fondée par les deux co-architectes du Zend Engine, et Facebook, utilisateur majeur de PHP, ont toutes deux lancé une initiative pour améliorer le langage.

PHPNG

Depuis le début de l'année 2014, Zend Technologies, dont le noyau Zend Engine II propulse toujours PHP, a démarré un projet nommé PHPNG (PHP Next Generation), conçu pour optimiser les performances de PHP en refactorisant le Zend Engine, tout en maintenant une compatibilité avec la version actuelle. Des benchmarks ont démontré de bien meilleures performances sur des projets PHP majeurs. L'avenir de PHPNG reste incertain quant à son implémentation comme remplaçant du Zend Engine. Les discussions entre les core contributeurs restent actives.

HHVM

A partir de 2008, Facebook, qui a depuis longtemps une utilisation massive de PHP pour répondre à ses besoins considérables, a développé une nouvelle implémentation de PHP, qui a changé de nom plusieurs fois. D'abord, HipHop for PHP, puis HPHPc, il s'agit d'un compilateur de code PHP qui transforme le code PHP en C++. Facebook a également développé HPHPi, qui est un interpréteur de code. Facebook a mis en Open Source ses différentes implémentations et les utilise pour ses propres besoins.

Conclusion

A ce jour, phpng a de grandes chances de devenir le nouveau cœur du langage pour une future version, mais d'autres acteurs peuvent être encore intervenir et proposer d'autres alternatives qui pourraient remporter les suffrages de la communauté PHP. Qui vivra verra !

Quelle que soit la décision prise par les core contributeurs, nous pouvons être sûrs que celle-ci suivra de toute façon les tendances du web ; les attentes des développeurs seront prises en considération pour ce choix. PHP n'a jamais déçu ses utilisateurs et ne va pas commencer à l'approche de sa vingtième année...

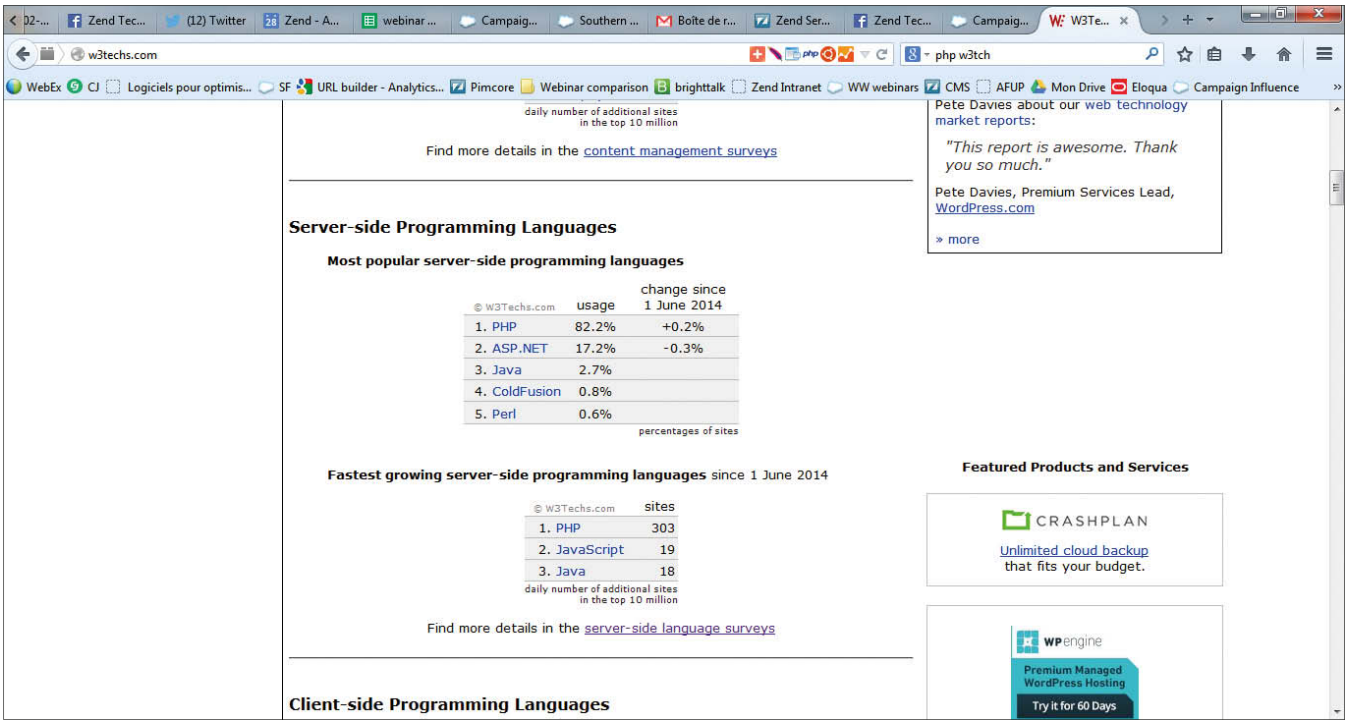
Depuis l'apparition de PHP, de nombreux projets Open Source ont vu le jour autour de ce langage. Le plus connu reste son logo et sa mascotte : 'un éléphant', dessiné en 1998 par Vincent Pontier (aka El Roubio).

Le choix de l'éléPHPant a été retenu pour de nombreuses raisons : Tout d'abord, tout le monde aime ce gentil animal et il rend bien des services à l'homme. De plus, il est à la fois puissant et docile. Enfin, il est rapide quand il attaque (une base de données). Bien entendu, les lettres PHP forment aussi un éléPHPant... Regardez bien l'image et vous verrez apparaître les trois fameuses lettres.

Depuis sa naissance, cette mascotte a trouvé sa place auprès des développeurs, développeuses et des associations comme l'AFUP (Association Française des Utilisateurs de PHP) qui ont lancé de nombreux goodies. L'engouement autour de l'éléPHPant est tel que de nombreux concours ont vu le jour comme le plus grand nombre de FreeHugs (câlins) avec l'éléPHPant pendant des salons

Christophe **Villeneuve**
Consultant IT pour Neuros, auteur du livre "Drupal 7 en mode avancé" aux éditions Eyrolles et auteur aux Editions ENI, Rédacteur pour WebRIVER, membre des Teams DrupalFR, AFUP, LeMug.fr, Drupagora...

Christophe **Chervy**
Responsable Marketing & Communication chez Zend Technologies Ltd.



Source : W3Tech – Juillet 2014 - http://w3techs.com/

Le futur de PHP

Après bientôt 20 ans d'existence et d'évolution permanente, et après avoir conquis la planète en devenant le langage le plus populaire du web, quel est désormais l'avenir de PHP ?

Ces jours-ci l'actualité se focalise sur la nouvelle version mineure PHP 5.6 mais ces derniers mois ont vu la naissance d'une nouvelle vague de piles PHP dédiées à la performance. Sous l'impulsion de Facebook et de sa HHVM, la communauté s'est en effet trouvé de nouvelles aspirations, faisant de la compilation « Just In Time » le nouvel Eldorado d'une communauté en manque d'innovation. Cet article vous propose de vous mettre à jour sur l'état de l'art de PHP depuis les nouveautés de PHP 5.6 jusqu'aux piles PHP du futur que sont HHVM et PHPNG.

PHP 5.6

Chaque version mineure de PHP 5 a fait progresser le langage : l'apport des traits et l'amélioration considérable des performances de PHP5.4, l'OP-Cache et les générateurs pour PHP 5.5. PHP 5.6 ne fait pas exception à la règle et cette nouvelle mouture de PHP 5, si elle ne brille pas par son apport en termes de performance, nous est livrée avec un lot de nouveautés.

PHPDBG, le débogueur intégré de PHP

Ce nouveau débogueur PHPDBG souhaite venir en remplacement du traditionnel X-Debug. Il apporte de nouvelles fonctionnalités et une empreinte sur les performances beaucoup plus faible que son prédécesseur. Contrairement aux débogueurs de la génération précédente, PHPDBG est agnostique en termes de SAPI, et on peut définir n'importe quel type d'environnement d'exécution à partir d'un fichier PHP d'amorçage. De la même façon on peut modifier les variables globales à la volée. La grande nouveauté apportée par PHPDBG est la flexibilité de définition des points d'arrêt. Il est en effet possible de positionner des points d'arrêt sur une ligne d'un fichier PHP, à l'entrée d'une classe, d'une méthode, d'une fonction ou encore sur une ligne de l'opcode. On peut aussi définir une condition à l'arrêt de l'exécution à un point particulier. La définition des points d'arrêts se fait par la ligne de commande de PHPDBG mais on dispose également d'une API PHP permettant de créer ou supprimer les points d'arrêts depuis le code PHP en cours d'exécution. Une fois positionné sur un point d'arrêt on a accès à différents commandes. On peut par exemple calculer et afficher n'importe quelle valeur en cours d'exécution comme le contenu d'une variable, d'une constante ou toute autre expression PHP. On peut par exemple afficher le « var_dump » d'une variable. Il est aussi possible d'avoir accès au code PHP ou à l'opcode en cours d'exécution, soit en spécifiant le nombre de lignes à afficher depuis le point d'arrêt soit en donnant le nom de la fonction, la classe ou la méthode désirée. Pour une analyse fine, la reprise de l'exécution depuis un point d'arrêt peut se faire ligne par ligne. A l'image de Zend Debugger, PHPDBG supporte aussi le débogage à distance. Il suffit d'indiquer à PHPDBG l'IP et les ports d'entrée et sortie, et vous pourrez déboguer sur le système distant. Pour ce type de débogage on peut aussi utiliser un client graphique développé en Java. A l'heure où j'écris ces lignes aucun IDE pour PHP ne propose PHPDBG comme débogueur intégré. Il faudra donc attendre quelques mois pour bénéficier pleinement des excellentes capacités de PHPDBG dans votre environnement de développement. Voyons maintenant les nouvelles fonctionnalités du langage en lui-même.

Support des expressions scalaires

Désormais PHP supporte des expressions scalaires pour la définition des constantes et les valeurs par défaut des propriétés et des paramètres de fonction. Une expression scalaire est une expression évaluée par le langage contenant des constantes, des scalaires et des opérateurs.

```
const UN = 1;
const DEUX = UN + 1;

function test($a = UN + 1, $b = '1 et 1 font ' . DEUX)
{
        if ($a == DEUX) return $b;
        else return 'Ne marche pas avec ' . $a;
}
echo test() ."\n";
echo test(3) . "\n";
```

On obtiendra

```
1 et 1 font 2
Ne marche pas avec 3
```

L'opérateur « ... » (splat operator)

Dans les versions antérieures de PHP, il était possible de définir des fonctions et méthodes dont le nombre de paramètres pouvait varier. Connaître le nombre de ces paramètres et en obtenir la valeur dans le corps de la fonction nécessitait l'utilisation de fonctions spéciales comme func_get_args() ou func_num_args(). Réciproquement, il était possible de passer un ensemble de n paramètres à une fonction dont le nombre de paramètre n'était pas connu, et cela au moyen d'un tableau. C'était le cas pour les méthodes magiques __invoke(), __call() ou la fonction call_user_func_array(). Désormais PHP 5.6 nous permet de définir ces ensembles de n paramètres au moyen du mot-clé « ... ».

Dans le cas de la définition d'une fonction le paramètre « ...$c » correspondra à un tableau $c dans le corps de la fonction, mais le passage de paramètre se fera comme s'il s'agissait d'une suite de n paramètres :

```
function variadic ($a, $b, ...$c){
        echo $a . "\n";
        echo $b . "\n";
        for ($i = 0; $i < count($c); $i++) {
                echo $c[$i] . "\n";
        }
}
variadic('a','b','c1','c2','c3');
```

On obtiendra

```
a
b
c1
c2
c3
```

Dans le cas inverse, on peut désormais passer un ensemble de n paramètres à une fonction quelconque en utilisant « ... » :

```
$tab = [1,2,3];
function sum ($a, $b, $c, $d)
{
        return $a + $b + $c + $d;
}
echo sum(1,...$tab);
```

Ce code nous donnera

```
7
```

Extension du mot-clé « use » aux fonctions et constantes

Le mot-clé « use » permettant de faire appel à un espace de nommage différent de celui dans lequel on travaille était jusqu'ici réservé aux classes. PHP5.6 étend cette notion aux fonctions et constantes :

```
//Fichier ns1.php
namespace ns1;
const MA_CONSTANTE = 1;

//Fichier ns2.php
namespace ns2;
use const ns1\MA_CONSTANTE;
function ma_fonction()
{
        echo ns1\MA_CONSTANTE;
}

//Fichier ns3.php
namespace ns3;
use function ns2\ma_function;

require_once 'ns2.php';
require_once 'ns1.php';

echo ma_fonction();
```

En exécutant le fichier ns3.php on obtiendra :

```
1
```

Autres nouveautés

Opérateur « puissance »

Désormais on peut utiliser l'opérateur ** pour utiliser la fonction « puissance » :

```
echo 2 ** 3;
```

va retourner : 8

Opérateurs mathématiques et les grands nombres entiers

Les entiers longs générés par la librairie GMP supportent désormais les opérateur mathématiques classiques : +, - , *, etc.

```
$a = gmp_init(40);
$b = gmp_init(20);
```

Avant PHP5.6

```
gmp_add($a, $b);
gmp_add($a, 17);
```

Avec PHP5.6

```
$a + $b;
$a + 17;
```

Autres

- PHP 5.6 accepte désormais des chargements de fichiers de 2 Go.
- La directive « default_charset » est désormais utilisée comme liste de caractère par défaut pour les fonctions dépendantes de l'encodage comme « htmlspecialchars() ».
- On trouve aussi une amélioration du support SSL/TLS et un enrichissement des fonctions liées au hachage et à l'encryptage de données.

Quel PHP pour demain ?

L'abandon de PHP 6 a permis de débloquer l'évolution de PHP sur un grand nombre de points. Mais les fonctionnalités envisagées à cette époque et qui ont été mises de côté restent à implémenter. Le retour de PHP 6 était donc inévitable, et la communauté des développeurs de PHP a d'ores et déjà débuté les premier travaux. Il est cependant trop tôt pour dire quel sera finalement le contenu de cette nouvelle version : si les anciennes spécifications de PHP 6 tiennent toujours, les récentes implémentations de PHP comme HHVM vont avoir un impact sur les décisions.

L'émergence de ces nouvelles implémentations de PHP a débuté il y a 6 ans. En 2008, Facebook doit faire face à un nombre d'utilisateurs toujours plus grand et le code initial en PHP n'est pas suffisamment performant. Pour cela il met en œuvre HPHPc un compilateur qui traduit le code PHP en code C++. Le gain de performances est énorme mais le temps de compilation est trop long et les développements perdent rapidement leur agilité. Afin de pallier à ces lacunes, Facebook met en production HHVM en 2011. Les performances sont équivalentes à celle d'HPHPc mais la compilation est réalisée à la volée par un compilateur JIT. Le gain de performance annoncé est impressionnant: 3 à 10 fois plus rapide que PHP 5.4. Depuis, la communauté PHP s'est emparée du projet HHVM et s'attache à rendre le compilateur JIT le plus compatible possible avec le langage PHP classique en intégrant PHP5.5 et en testant le plus possible de frameworks et d'applications communes en PHP. Malgré tout, non seulement la compatibilité n'est pas encore totale, notamment pour les deux frameworks les plus populaires, mais les performances annoncées initialement par Facebook ne sont toujours pas au rendez-vous. C'est que l'accélération de l'exécution du code induite par la compilation ne fait pas tout, et d'autres facteurs influent sur la performance générale des processus PHP. Ainsi, au début du mois de mai de cette année, Dmitri Stogov de Zend Technologies a fait part d'une expérience menée sur le noyau du Zend Engine, dans le but d'améliorer de façon significative la rapidité d'exécution et la consommation mémoire des processus PHP. Cette « nouvelle génération » de PHP (PHPNG) basée sur une refonte du moteur actuel de PHP, se présente comme un nouveau champ de recherche pour la course à la performance désormais engagée. L'idée est d'optimiser l'allocation des données en mémoire non seulement afin d'accélérer l'exécution des scripts PHP, mais aussi de préparer le Zend Engine actuel à se munir dans le futur d'une compilation JIT très efficace. Cette approche offre l'avantage de limiter les efforts à fournir pour rendre compatibles les extensions actuelles avec le nouveau noyau.

Les premiers tests font état d'une remarquable performance de PHPNG vis-à-vis de HHVM. Même si HHVM reste le moteur le plus rapide, il se présente de nombreux cas où les résultats sont plutôt désastreux. PHPNG améliore les processus dans tous les cas et le gain de performance est comparable à celui d'HHVM.

	PHP 5.6	HHVM 3.1	PHPng
Wordpress 3.6.0	22 req/s	+53 %	+30 %
ZF1 Hello world	1175 req/s	-17 %	+28 %
Drupal 6.13	1629 req /s	-16 %	+15 %

En conclusion de cet article, nous pouvons dire que l'avenir de PHP semble donc se concentrer plus particulièrement sur la performance pure. Face aux langages concurrents que sont dotNet et Java, cela ne peut être que bénéfique. Car si PHP est devenu le langage incontournable des applications web, la pénétration de PHP au sein des entreprises pour les applications critiques reste encore marginale.

Acquérir un niveau de performance plus élevé est donc un argument de plus pour valoriser l'utilisation de PHP au sein des grandes entreprises. PHP 5.6 ne sera pas une grande révolution, mais les propositions pour la prochaine version majeure nous promettent un PHP rapide et sûrement plus professionnel encore. Un grand merci à Pascal MARTIN pour son aide précieuse à la rédaction de cet article.

Sophie **BEAUPUIS**
Consultante PHP chez Zend Technologies.

PHP en mode cloud : déployer une application PHP sur Gandi Simple Hosting

Gandi Simple Hosting est un service d'hébergement de type PaaS (Platform as a Service) pensé pour les développeurs. La plateforme supporte actuellement PHP, Node.js, Python et Ruby et les bases de données MySQL, PostgreSQL et MongoDB. Il permet de déployer et de gérer son code avec Git, SSH, sFTP et dispose également d'une API XML-RPC.

Dans cet article, nous allons déployer une application PHP sur Simple Hosting de la manière suivante:

- Le code de l'application fonctionnera localement sur notre ordinateur
- Nous le déploierons sur un site de *staging* sur notre instance Simple Hosting, avec Git et SSH
- Après nos tests, nous déploierons l'application sur un site de *production*, sur la même instance Simple Hosting

Nous utiliserons le squelette de base d'une application écrite avec Yii, un puissant framework MVC (Model – View – Controller) en PHP, disponible sous licence «new BSD» et donc utilisable pour des applications libres ou propriétaires.

Pré-requis

Nous avons besoin d'installer PHP 5, Git et SSH sur notre machine de développement. Sous Windows, nous pouvons installer :

- l'exécutable de PHP 5, disponible sur http://windows.php.net/download/. Il est nécessaire pour générer le squelette de notre application (les versions Thread Safe et Non Thread Safe fonctionnent).
- Git pour Windows. Beaucoup d'options existent, mais nous suggérons **msysgit**, (disponible sur http://msysgit.github.io/) qui propose un client SSH, une console Git et une interface graphique.

Après avoir téléchargé PHP, nous décompressons le contenu dans `c:\program files\php` et ajoutons le répertoire au *PATH* de notre système (Panneau de Configuration > Variables Système).

Vérifions que PHP est correctement installé en ouvrant la ligne de Commandes et en exécutant `php -v`

```
C:\>php -v PHP 5.5.14 (cli) (built: Jun 25 2014 12:41:10)
Copyright (c) 1997-2014 The PHP Group Zend Engine v2.5.0,
Copyright (c) 1998-2014 Zend Technologies
```

Pour installer Git, il nous suffit de lancer le programme d'installation que nous avons téléchargé et de suivre les instructions.

Sous Linux, et en fonction de notre distribution:

- Ubuntu ou Debian: `sudo apt-get install php5-cli git-core openssh openssh-client`
- Fedora, Centos, ou Red Hat: `sudo yum install php-cli git-core openssh openssh-clients`

Création de l'instance Simple Hosting

Nous pouvons sélectionner et créer notre instance sur www.gandi.net/hosting/simple. Si nous avons déjà un domaine chez Gandi, nous bénéficions de 5 jours d'essai gratuit et 50% de réduction sur une instance, ainsi que d'un certificat SSL gratuit.

Les instances sont disponibles en plusieurs tailles (du S au XXL), équivalentes à la puissance nécessaire pour servir plus ou moins d'utilisateurs. Les certificats SSL peuvent être utilisés pour encrypter les requêtes sur les instances de taille M ou supérieure. Notre instance sera hébergée au datacenter de Bissen, au Luxembourg, au coeur de l'UE – nous aurions aussi pu

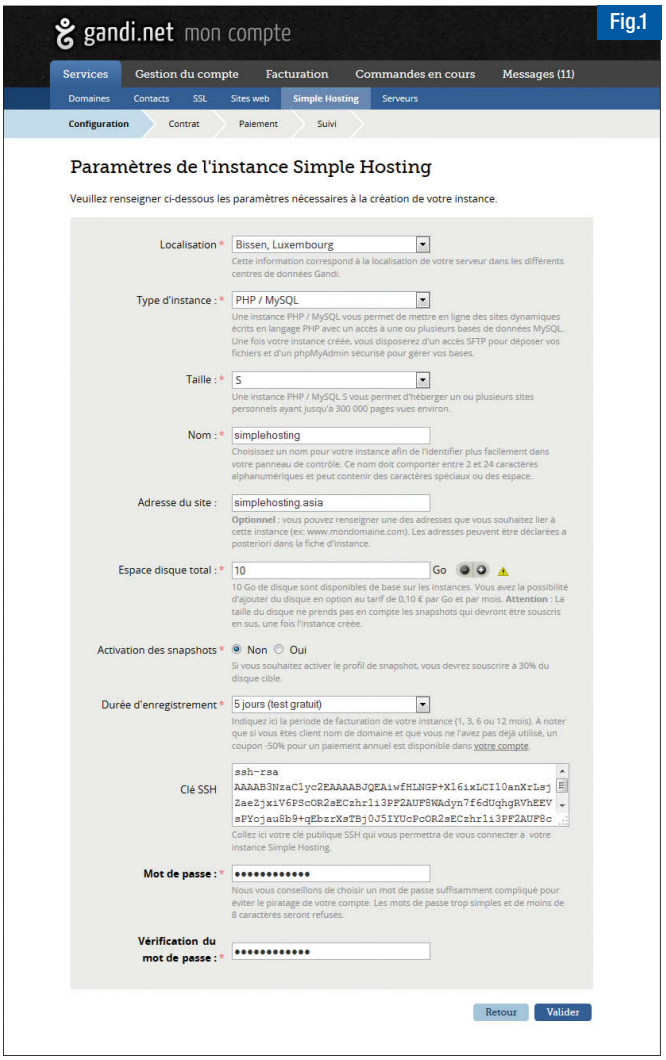


Fig.1

Création de l'instance Simple Hosting

choisir Paris (France) ou Baltimore (USA) Fig.1.

Le champ "Site web" n'est pas obligatoire puisqu'il est possible d'en ajouter plus tard et que Gandi nous offre, par défaut, une adresse de test pour accéder à notre instance.

En plus du mot de passe, nous fournissons également notre clé SSH publique pour permettre l'authentification sécurisée sans mot de passe (plus d'informations disponibles sur http://wiki.gandi.net/fr/simple/ssh_key)

Une fois notre instance créée, elle sera disponible dans notre compte Gandi comme montré dans la Fig.2.

Environnements et VHOSTS

Avec les instances PHP de Simple Hosting, nous pouvons traiter chaque VHOST comme un environnement ou site différent. Les différentes bases de code sont hébergées sur la même instance, mais de manière isolée.

- Chaque VHOST est associé à un dépôt Git hébergé sur l'instance. Ce dépôt est accessible via *git+ssh* et une URL fournie par Gandi.
- Nous pouvons également créer plusieurs bases de données avec différents utilisateurs au sein d'une même instance

Nous allons utiliser l'adresse de test fournie par Gandi (*27eebd9d1b.testurl.ws*, dans notre exemple) pour notre environnement de *staging*, et notre nom de domaine (*simplehosting.asia* dans ce cas) pour notre environnement de *production*.
Pour nous servir de Git et SSH pour le déploiement de l'application, nous allons utiliser l'information disponible dans le tableau des connexions, comme indiqué dans la Fig.3.
Par exemple, notre environnement de *staging* sera disponible sur

```
git+ssh://49309@git.dc2.gpaas.net/27eebd9d1b.testurl.ws.git
```

Et notre environnement de *production* sera disponible sur

```
git+ssh://49309@git.dc2.gpaas.net/simplehosting.asia.git
```

Création et configuration de l'application

Nous commençons par créer un répertoire vide qui va contenir notre application, par exemple "myyii". Après voir téléchargé Yii (version 1.1.15) depuis http://www.yiiframework.com/download/ , nous pouvons en extraire les contenus et les placer dans un nouveau répertoire "myyii". Profitons-en également pour renommer le répertoire « Yii » en "yii".
Puis créons aussi un répertoire "htdocs" à la racine de l'application.
La structure devrait ressembler à la suivante : Fig.4.

```
myyii/ myyii/yii/demos myyii/yii/framework myyii/yii/requirements
myyii/htdocs
```

Sous Windows et avec **msysgit**, nous pouvons lancer *Git Bash* depuis l'Explorateur Windows avec le bouton droit de la souris. Pour la suite, nous devons simplement nous assurer que nous sommes à la racine de "myyii" pour exécuter les commandes. Pour commencer, identifions-nous pour que Git puisse signer nos *commits*:

```
git config --global user.name "Notre nom complet ou pseudo"
git config --global user.email nous@example.com
```

Sous Windows, nous allons désactiver CRLF (carriage-returns-line-feed)

```
git config --global core.autocrlf=false
```

Créons maintenant notre dépôt pour gérer le code de notre application avec Git.

```
git init .
```

Désormais, nous sommes prêt pour générer le squelette de notre application Yii.

Fig.2

Instance Simple Hosting dans notre Compte Gandi

```
php yii/framework/yiic.php webapp htdocs git
```

Terminons notre premier *commit,* en ajoutant d'abord les fichiers à notre arbre git:

```
git add yii git add htdocs git commit -m "First version"
```

Gestion du code et déploiement

Pour faire des actions, il faut ajouter deux *remote*, correspondant à nos deux environnements. Pour *staging* :

```
git remote add staging git+ssh://49309@git.dc2.gpaas.net/
27eebd9d1b.testurl.ws.git
```

Et pour *production* :

```
git remote add production git+ssh://49309@git.dc2.gpaas
.net/simplehosting.asia.git
```

Notre instance Simple Hosting héberge nos dépôts Git séparément du code de notre application. Ainsi, quand nous mettons à jour le dépôt avec Git, l'application n'est pas immédiatement mise à jour.
Pour la mettre à jour , nous utilisons la commande *deploy,* via SSH, qui va :

- copier les contenus de la branche *master* dans le conteneur de notre application (les fichiers créés par vos utilisateurs ou votre application restent intactes)

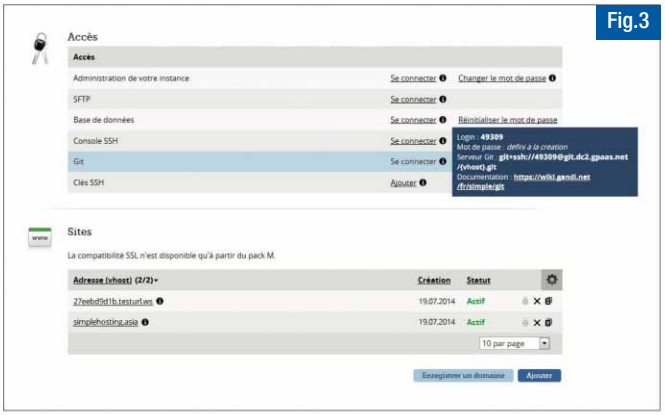


Fig.3

Gestion de l'instance Simple Hosting

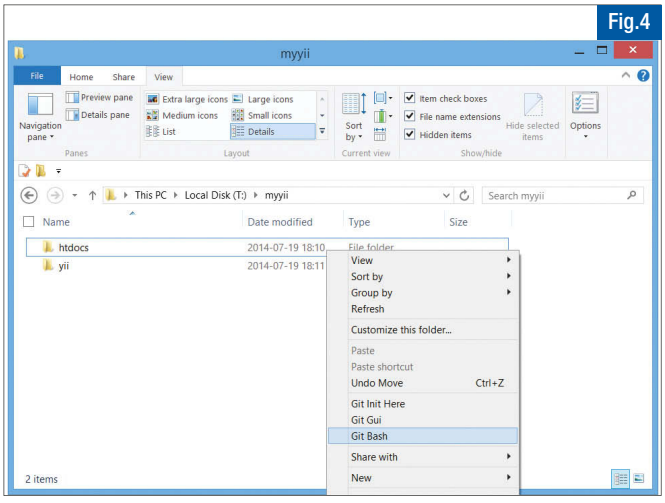


Fig.4

Git Bash et structure de notre application sous Windows

- exposer le répertoire *htdocs* au public (les autres ressources de l'application restent privées)

La branche *master* est la branche par défaut de Git sur Simple Hosting et notre recommandation est de la maintenir prête à l'usage en *production*.
Nous ferons nos développements incrémentaux dans des branches dédiées (comme *version-001, feature-X* ou *staging*) pour les tester localement ou sur notre environnement de staging avant de les fusionner avec *master*, comme plus loin dans notre exemple.
Notre objectif est donc de déployer notre code de la manière suivante sur notre environnement de *staging* :

```
# pour mettre à jour la branche master de notre dépôt sur le VHOST de test  # avec les contenus de la branche version-001
git push staging version-001:master      # pour déployer l'application     ssh 49309@git.dc2.gpaas.net deploy 27eebd9d1b.testurl.ws.git
```

Et en *production* :

```
# pour mettre à jour la branche master du dépôt de production
git push production master
# pour déployer l'application ssh 49309@git.dc2.gpaas.net deploy simplehosting.asia.git
```

Déployer l'application en *staging*

Pour créer et travailler dans une branche spécifique à partir de la branche master, il suffit d'exécuter la commande suivante :

```
git checkout -b staging_master
```

Après avoir modifié notre code, nous le sauvegardons avec :

```
git add .        git commit -m 'Improvements on something'
```

Enfin nous pouvons tester nos modifications sur notre site de *staging*, en utilisant "*git push*" puis en exécutant la commande "*deploy" en SSH.*

```
git push staging staging_master:master ssh 49309@git.dc2.gpaas.net deploy 27eebd9d1b.testurl.ws.git
```

Nos modifications sont maintenant visibles sur http://27eebd9d1b.testurl.ws/ **Fig.5**.

Déployer l'application en *production*

Après avoir vérifié que nos modifications fonctionnent sur *staging*, nous pouvons intégrer nos changements à la branche *master* et les déployer en production. Nous retournons sur la branche *master* et y ajoutons les changements faits dans *staging_master*.

```
git checkout master git merge staging_master
```

Une fois nos changements inclus dans *master*, nous pouvons les rendre disponibles à nos utilisateurs sur notre site en *production*

```
git push production master  ssh  49309@git.dc2.gpaas.net deploy simplehosting.asia.git
```

Et voilà, nos modifications sont visibles sur notre site public !

Pour aller plus loin

- Utilisez l'interface PHPMyAdmin de Simple Hosting pour créer et gérer vos bases de données: http://wiki.gandi.net/en/simple/mysql
- Configurez Yii dans `htdocs/protected/config/main.php`
- Essayez différentes extensions pour Yii: http://www.yiiframework.com/extensions/
- Optimisez la performance de votre application avec l'Accélérateur Web fourni par Simple Hosting (http://wiki.gandi.net/fr/simple/cache) couplé au cache applicatif de Yii (http://www.yiiframework.com/doc/guide/1.1/fr/caching.overview)
- Montez en charge et supportez plusieurs millions de pages vues mensuelles avec les packs L, XL et XXL https://www.gandi.net/hosting/simple

Alex 'Peck' Solleiro - *Gandi*

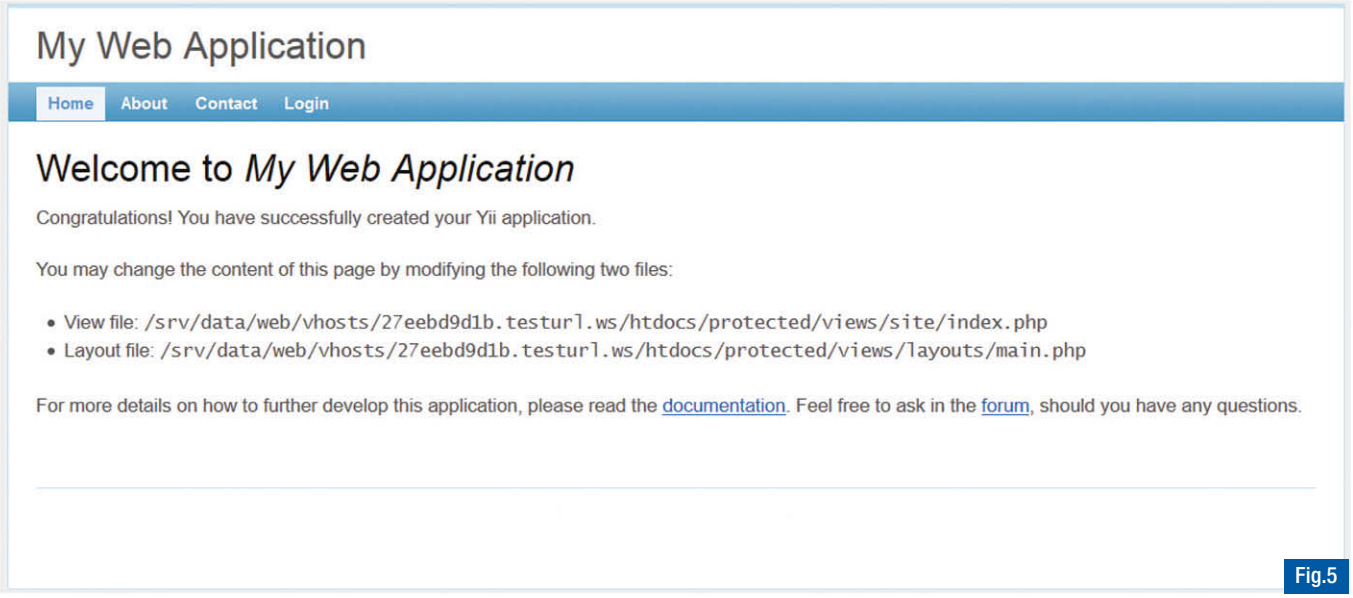


Fig.5

Application déployée sur l'environnement de staging

Créer rapidement un projet php avec Symfony 2

Pour cet article nous partons du principe que vous avez un environnement Apache, PHP, Mysql fonctionnel. Celui-ci a donc pour but d'installer Symfony, de découvrir ce qu'il contient et de créer notre premier Bundle. L'idée est de créer un petit site collaboratif, dans lequel les utilisateurs pourront ajouter des recettes de cuisine et mettre à disposition les sources du projet sous licence libre sur Github.

Installer symfony

Remarque : l'ensemble des commandes données sont faites pour être exécutées en environnement Unix-Like (Linux ou OS X), pour les utilisateurs de Windows, à vous d'adapter les commandes. Nous travaillons pour l'exemple dans le répertoire /home/programmez-symfony/symfony , pensez à adapter ce chemin. Pour installer Symfony, nous avons choisi d'utiliser la méthode préconisée à savoir utiliser Composer. Nous commençons donc par installer Composer :

```
$ curl -s https://getcomposer.org/installer | php
```

Composer est un fichier exécutable, nous l'utilisons pour télécharger la version standard de Symfony 2

```
$ php composer.phar create-project symfony/framework-standard
-edition /home/programmez-symfony 2.4.4
```

Pour que notre site réponde sur l'adresse : http://www.programmez-symfony.net nous devons créer et renseigner le domaine dans notre fichier /etc/hosts. Ouvrez le fichier et remplacer la ligne :

```
127.0.0.1 localhost
```

par

```
127.0.0.1 localhost www.programmez-symfony.net
```

Puis créer un fichier VirtualHost dans /etc/apache2/sites-available/programmez-symfony.net

```
<VirtualHost *:80>
  ServerName www.programmez-symfony.net
  ServerAlias programmez-symfony.net
  DocumentRoot /home/programmez-symfony/symfony/web/
  DocumentIndex app.php
</VirtualHost>
```

Nous activons le site avec la commande

```
$ a2ensite programmez-symfony.net
```

et redémarrons apache

```
$ /etc/init.d/apache2 restart
```

Nous avons donc un Symfony opérationnel, qui appelle bien la page de base de Symfony qui est app.php via l'url http://www.programmez-symfony.net
Nous allons vérifier que notre serveur est bien configuré en allant sur la page http://www.programmez-symfony.net/config.php
Vous arrivez sur une page de ce type **Fig.1.**
Vous voyez alors deux types de message :

- Major problems : à corriger impérativement, comme par exemple des problèmes de droits sur les répertoires
- Recommandations : des recommandations pour optimiser le fonctionnement de Symfony, à suivre si possible, mais non bloquant.

Une fois nos problèmes résolus (ici des droits d'écriture à donné à Apache et une timezone à définir), nous pouvons maintenant démarrer notre projet, si nous allons à la page de débug : http://www.programmez-symfony.net/app_dev.php, nous obtenons la page suivante : **Fig.2.**
La page par défaut (app.php) affiche une erreur 404, c'est normal, nous n'avons pas encore dit à Symfony ce qu'il devait afficher.

Fig.1

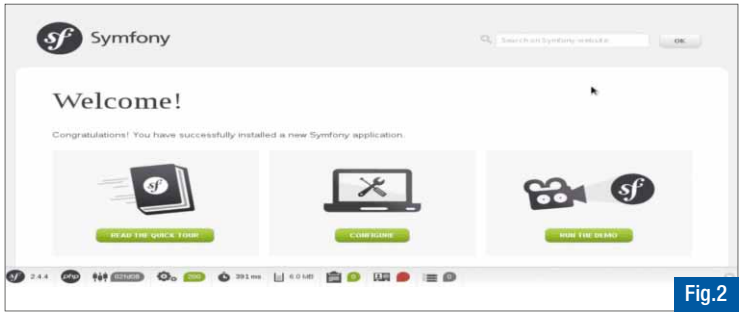


Fig.2

Page d'accueil de Symfony

Découverte du contenu de symfony

L'arborescence de Symfony est composée ainsi :

- app : contient l'intégralité de la configuration de notre site (connexion à la BDD, définition des routes, sécurité, etc.),
- web : la partie publique de notre site, avec par exemple les images et feuilles de styles qui doivent pouvoir être chargées,
- bin : les fichiers nécessaires à l'installation des vendors,
- src : contient tous les fichiers sources, c'est principalement dans ce dossier que nous allons travailler, les sous-dossiers :
  - Application (il peut y avoir plusieurs sous-dossiers)
    - Bundle (un bundle est une fonctionnalité de notre site comme les recettes ou les utilisateurs)
      - Entity : modèle de données
      - Controller : contrôleur contenant notre code métier
      - Repository : Méthodes spécifiques à l'application pour gérer les donnnées
      - Ressources
        - View : les fichiers d'affichage de notre bundle
        - Config : la configuration des routes notamment
      - Tests : pour placer les tests unitaires et fonctionnels de notre bundle
- vendor : les librairies utiles pour notre projet spécifiques à Symfony (Doctrine, Twig, etc).

Création de notre premier Bundle

Pour créer un bundle nous pouvons le faire à la main en créant les fichiers et en les déclarant à Symfony, mais le plus simple et le plus rapide est d'utiliser le générateur de Bundle pour cela nous utilisons la commande :

```
$ php app/console generate:bundle -namespace="ProgrammezSymfony
/RecetteBundle" --bundle-name="ProgrammezSymfonyRecetteBundle"
 --dir="src" --format="annotation"  --structure
```

On demande à Symfony de générer un bundle avec pour namespace ProgrammezSymfony/RecetteBundle et pour nom ProgrammezSymfonyRecetteBundle. Il crée ce bundle dans le répertoire « src » qui utilise le format « annotation » et l'option structure, permet de générer l'intégralité des répertoires mêmes ceux qui sont vides. A la validation de la commande Symfony nous demande de confirmer la création du Bundle et de confirmer deux mises à jour, nous avons choisi yes dans tous les cas. Nous avons donc tout le nécessaire pour démarrer le travail sur notre Bundle.

Remarque : le nom de votre bundle doit obligatoirement finir par Bundle pour que Symfony le détecte.

En détail Symfony a fait les actions suivantes :

- Déclarer le bundle à Symfony en éditant le fichier app/appKernel.php et en déclarant le nouveau Bundle dans la fonction registerBundles
- Indiquer à Symfony d'utiliser notre Bundle lorsque nous arrivons sur la page d'accueil du site en modifiant le app/config/routing.yml

Nous avons donc créé notre premier bundle que nous pouvons appeler en utilisant l'adresse (indiquée dans /src/ProgrammezSymfony/RecetteBundle/Controller/DefaultController.php) http://www.programmez-symfony.net/app_dev.php/hello/toto Vous aurez alors une page affichant simplement Hello Toto.

Création et affichage de notre modèle de données avec les entités de Symfony

Pour gérer les données avec une base de données, Symfony s'appuie sur l'ORM Doctrine. C'est lui qui va s'occuper de créer la base, les tables, et de gérer l'insertion, la sélection, la suppression de données. Symfony utilise les entités. Nous allons donc utiliser la génération des entités pour créer notre première entité :

```
$ php app/console doctrine:generate:entity -entity="Programmez
SymfonyRecetteBundle:Recette"   -fields="name:string(255)
tpspreparation:integer tpscuisson:integer ingredients:text
preparation:text"
```

Symfony vous pose une série de questions pour générer notre entité, nous avons choisi de laisser les options par défaut.
Nous demandons donc à doctrine de générer une entité (doctrine:generate:entity), le paramètre –entity permet de donner le nom du bundle, puis après les deux points le nom de l'entité, et l'option –fields permet de définir nos champs avec un nom de champ et un type.
Si nous regardons dans le fichier /src/ProgrammezSymfoy/RecetteBundle/Entity/Recette.php, nous voyons les champs déclarés sous forme d'annotation comme par exemple le champs tpscuisson :

```
/**
* @var integer
*
* @ORM\Column(name="tpscuisson", type="integer")
*/
private $tpscuisson ;
```

Symfony a donc généré la déclaration des variables pour nous ainsi que les getters et setters.
Pour générer notre base et les tables nous utilisons deux commandes :

```
$ php app/console doctrine:database:create
$ php app/console doctrine:schema:create
```

En cas de modification des champs ou d'ajout d'autres entités, vous pouvez demander à doctrine la mise à jour de la BDD avec la commande suivante :

```
$ php app/console doctrine:schema:update -force
```

Pour continuer, nous renseignons manuellement dans la base quelque données, nous aurions pu aussi utiliser Fixtures, pour cela, je vous laisse aller voir la doc officielle :
http://symfony.com/fr/doc/current/bundles/DoctrineFixturesBundle/index.html

Affichage de nos recettes

Pour l'affichage de nos pages, Symfony utilise le moteur de template Twig. Nous allons donc commencer par regarder comment est affichée notre page Hello Toto. Le fichier appelé est dans /src/ProgrammezSymfony/RecetteBundle/Resources/views/Default/index.html.twig
Ce fichier contient simplement :

```
Hello {{name}}
```

Nous allons ajouter un peu de HTML autour pour mettre en forme notre affichage et rendre notre fichier index générique, pour que l'intégralité des autres affichages hérite de l'index.
Puis on crée le fichier qui contiendra la liste de nos recettes /src/ProgrammezSymfony/RecetteBundle/Resources/views/Default/recettes.html.twig qui contient0 le code suivant :

```
{% extends 'ProgrammezSymfonyRecetteBundle:Default:index.html.twig' %}

{% block contenu %}
 {%for recette in Recettes %}
 <div class="fiche-recette">
   <div class="nom"> {{recette.name}}</div>
   <div class="ingredients"> {{recette.ingredients}} </div>
   <div class="preparation"> {{recette.preparation}} </div>
 </div>
 {% else %}<div class="alert">Désolé aucune recette pour
l'instant</div>
 {% endfor %}
{% endblock %}
```

Nous allons ensuite dire au contrôleur d'utiliser notre template index dans /src/ProgrammezSymfony/RecetteBundle/Controller/DefaultController.php nous remplaçons la ligne

```
@Route('/hello/{name}')
```

par

```
@Route('/')
```

Puis dans la fonction index située dans le même fichier, nous supprimons la variable $name qui n'est plus utile. La fonction devient alors :

```
public function indexAction() {
 return array() ;
}
```

Lorsque nous arrivons sur notre site nous avons alors une page vide qui contient pourtant bien la structure HTML que nous avons créée.
Voyons maintenant comment récupérer les informations depuis la base de données pour permettre leur affichage. Nous allons utiliser les fonctions

fournies par Doctrine avec la méthode findAll() ;.
Pour cela nous modifions notre fonction indexAction comme suit :

```
public function indexAction() {
  $repository  =  $this->getDoctrine()->getRepository('Programmez
SymfonyRecetteBundle:Recette');
    $variables['Recettes'] = $repository->findAll();

    return  $this->render('ProgrammezSymfonyRecetteBundle:
Default:recettes.html.twig', $variables);
}
```

Fig.3.

Nous avons donc vu l'affichage de nos recettes, mais il n'est pas très pratique de passer directement par des requêtes SQL ou phpMyAdmin pour insérer des données. Nous allons donc voir comment gérer des formulaires d'ajout sur notre site.

Création d'un formulaire d'ajout de recette

Afin de nous faciliter la vie, Doctrine a (encore) prévu un générateur de formulaire. Pour cela nous allons utiliser la commande :

```
$php app/console generate:doctrine:form  ProgrammezSymfony
RecetteBundle:Recette
```

Cette commande génère un formulaire à partir des informations de l'entité, n'hésitez pas à aller voir le code généré dans \src\ProgrammezSymfony\RecetteBundle\Form\RecetteType.php
Dans ce code vous trouverez différentes fonctions :

- buildForm, permet de déclarer les champs du formulaire
- setDefaultOptions : définit les options par défaut des champs. Par défaut le générateur reproduira les valeurs par défaut de l'entité
- getName : définit le nom du formulaire.

Ce code s'il n'est pas parfait, permet d'avoir un formulaire fonctionnel.
Nous allons donner le chemin de notre formulaire à Symfony, pour cela nous avons rajouté les lignes suivantes au routing.yml :

```
admin_recette_add:
  path:      /ajouter_recette
  defaults:  {  _controller:  ProgrammezSymfonyRecetteBundle:
Default:ajouter }
```

Puis nous allons définir dans notre contrôleur (DefaultController.php) l'action ajouter appelé par notre routeur :

```
    public function ajouterAction()
{
  $recette = new Recette;
  $form = $this->createForm(new RecetteType, $recette);
  $request = $this->get('request');
  if ($request->getMethod() == 'POST') {
    $form->bind($request);
    if ($form->isValid()) {
      $em = $this->getDoctrine()->getManager();
      $em->persist($recette);
      $em->flush();
       return new Response('Recette ajoutée');
    }
  }
  return $this->render('ProgrammezSymfonyRecetteBundle:Default:
ajout-recette.html.twig', array(
    'form' => $form->createView(),
  ));
}
```

Sans oublier dans ce même fichier de lui donner les classes à utiliser avec (en haut du fichier avec les autres appels) :

```
use Symfony\Component\HttpFoundation\Response;
use ProgrammezSymfony\RecetteBundle\Entity\Recette;
use ProgrammezSymfony\RecetteBundle\Form\RecetteType;
```

Il nous reste que l'affichage à gérer.

Affichage de notre formulaire

Une fois le formulaire généré, il nous reste à créer la vue associée. Pour cela nous créons un fichier ajout-recette.html.twig dans le répertoire /src/ProgrammezSymfony/RecetteBundle/Resources/views/Default/.
Ce fichier contient le code suivant :

```
{%  extends  'ProgrammezSymfonyRecetteBundle:Default:index.
html.twig' %}
{% block title %}Administration | Ajout d'une recette{% endblock %}

{% block contenu %}
<h1>Ajout d'une nouvelle recette</h1>
<form action="{{ path('admin_recette_add') }}" method="post"
 {{ form_enctype(form) }}>
    {{ form_widget(form) }}
    <input type="submit" />
</form>
{% endblock %}
```

Lorsque nous affichons notre formulaire, voici la page que nous obtenons : Fig.4.

Cet article avait pour but de vous donner de quoi démarrer rapidement un projet Symfony. Il ne vous reste plus qu'à coder, encore et encore avec Symfony pour connaître ce puissant framework.

Vanessa **Kovalsky David**
Lead développeuse Drupal / PHP chez Webnet - http://vanessakovalsky.net
Auteur du livre 'Drupal 7 en mode avancé' aux éditions Eyrolles
Membre actif de DrupalFr, et de la promotion du logiciel libre avec l'ALDIL
Lead dev sur le projet Esecouristes : https://github.com/vanessakovalsky/esecouristes

Fig.4

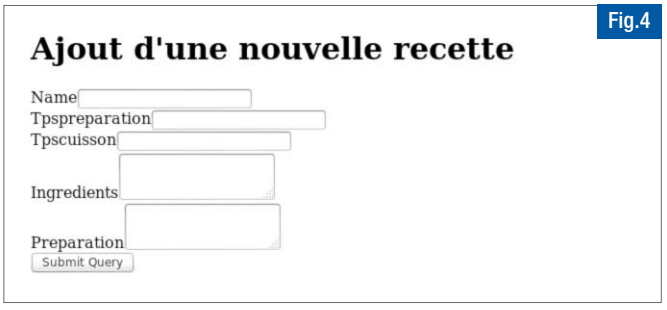
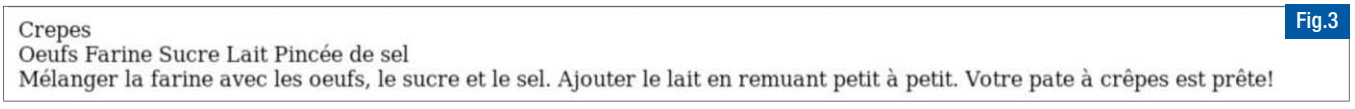


Fig.3



Affichage de notre première recette

PHP & GIT : le duo de choc

GIT est un contrôle de version décentralisé (ou DSCL pour Distributed Source Code Management). Il fut créé par Linus Torvalds. Il s'agit d'un outil qui se veut simple et performant et s'utilise sur le principe des serveurs décentralisés, ce qui le différencie des autres logiciels de versionning, basé sur un serveur centralisé. De nombreuses fonctionnalités sont proposées et il vous permet de suivre et de gérer l'historique de l'ensemble des fichiers et dossiers d'un projet open source, composé d'un ou plusieurs développeurs.

Le but de GIT est de stocker le contenu du fichier en BLOB (binary large object). Chaque dossier est représenté comme des arbres. Chaque arbre contient d'autres arbres (sous-dossiers), avec un fichier BLOB qui contient le mode, le type, le nom et un algorithme SHA-1 (généré automatiquement) pour chaque entrée de l'arbre. Bien entendu, lors des transferts, si le logiciel GIT rencontre plusieurs fichiers portant le même nom de contenu, ils seront différenciés.

Installer GIT

Pour utiliser GIT, vous devez installer le logiciel ou le paquet suivant votre système d'exploitation. Sous linux, vous devez installer le paquet GIT. Par exemple si vous utilisez Debian ou Ubuntu, il faut faire :

```
$ apt-get install git-core
```

Bien entendu GIT propose les différentes méthodes d'installation suivant la version de linux à partir de l'adresse suivante : http://git-scm.com/download/linux. Sur Mac, vous pouvez utiliser le logiciel MacPorts (http://www.macports.org) et exécutez la commande suivante :

```
sudo port install git-core
```

Sous Windows, la méthode est un peu plus longue, car vous devez générer une clé SSH. Pour cela, il vous faudra installer les logiciels Putty et Puttygen qui généreront la clé dont vous aurez besoin lors d'un déploiement vers un serveur externe. Ensuite vous devez télécharger GIT à partir de l'URL suivant : http://git-scm.com/download/win

Démarrer avec GIT

Après avoir installé GIT, vous devez commencer par créer un dépôt, mais vous devez prévoir l'ajout et la modification de ce dépôt.

Créer un Répository

Cette étape est nécessaire, surtout si vous êtes plusieurs sur un même projet, car il permet d'identifier les personnes qui vont participer au projet et commiter. Vous ouvrez un terminal pour créer un nom et un email :

```
$ git config --global user.name "Prenom Nom"
$ git config --global user.email "votreEmail"
```

Vous devez créer un répertoire qui contiendra tous les fichiers du projet et initialiser celui-ci pour une utilisation avec GIT

```
$ mkdir votre_projet
$ cd votre_projet/
```

Dans ce dossier, vous devez taper ceci :

```
git init
```

Fig.1

```
hello@hello-1215B-1215B: ~/www/gitphp
hello@hello-1215B-1215B:~/www/gitphp$ git status
# On branch master
#
# Initial commit
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#       index.php
nothing added to commit but untracked files present (use "git add" to track)
hello@hello-1215B-1215B:~/www/gitphp$
```

Vous obtenez le résultat suivant :

```
Initialized empty Git repository in /home/sean/new_project/.git/
```

Le message indique que votre dépôt est prêt à l'emploi et qu'il est vide.

Ajouter des fichiers

Pour versionner des fichiers PHP, vous créer un projet comme vous en avez l'habitude. Nous créons un fichier index.php que nous sauvegardons après :

```
<?php
echo 'Hello World<br />' ;
?>
```

Par ailleurs, il est possible de vérifier le statut de Git et si notre fichier est bien présent dans le dossier avec la fonction suivante : git status (Fig.1). La branche par défaut Git a bien pris en compte notre fichier créer. De plus, le statut fournit les informations à effectuer par rapport à vos fichiers. Ici, vous pouvez voir la ligne 'git add', ce qui correspond à versionner les fichiers de cette branche et se traduit pour notre exemple de la façon suivante : Fig.2

```
$ git add index.php
```

Nous pouvons voir que le statut de notre fichier a changé et par conséquent notre fichier 'index.php' est bien suivi par Git.

Commiter

Maintenant, vous pouvez commiter votre fichier et par la même occasion effectuer une image 'snapshot' de votre projet. Pour cela, nous devons alimenter correctement le journal de Logs pour garder un historique de cette opération. Le moyen le plus rapide est d'utiliser l'option -m pour saisir le message, de la manière suivante :

```
$ git commit index.php -m 'premier commit'
```

pour obtenir le résultat suivant (Fig.3). De plus, si vous vérifiez, vous verrez qu'il n'y a plus de fichiers en attente (Fig.4).

Journal des Logs

Le journal des logs permet d'avoir une trace des différentes opérations.

Fig.2

```
hello@hello-1215B-1215B: ~/www/gitphp
hello@hello-1215B-1215B:~/www/gitphp$ git status
# On branch master
#
# Initial commit
#
# Changes to be committed:
#   (use "git rm --cached <file>..." to unstage)
#
#       new file:   index.php
#
hello@hello-1215B-1215B:~/www/gitphp$
```

Fig.3

```
hello@hello-1215B-1215B: ~/www/gitphp
hello@hello-1215B-1215B:~/www/gitphp$ git status
# On branch master
#
# Initial commit
#
# Changes to be committed:
#   (use "git rm --cached <file>..." to unstage)
#
#       new file:   index.php
#
hello@hello-1215B-1215B:~/www/gitphp$ git commit index.php -m 'premier commit'
[master (root-commit) 6ff9337] premier commit
 1 file changed, 5 insertions(+)
 create mode 100644 index.php
hello@hello-1215B-1215B:~/www/gitphp$
```

Pour obtenir les informations du journal, nous exécutons la ligne de commande suivante : Fig.5

```
$  git log
```

Le résultat montre les informations de la personne qui a commité, la date, mais aussi les noms des fichiers et le message d'informations et de sa clé (au format SHA1).

Les mises à jour

Comme tous projets, aucun ne se réalise en une seule journée, et c'est pour cela, que vous devez aussi connaître les fonctions de mises à jour pour GIT. Nous allons modifier le fichier 'index.php', créé précédemment pour obtenir le code suivant :

```
<?PHP
echo "Hello World<br />";
echo "(c) Programmez 2014<br />";
?>
```

et nous vérifions le comportement de git avec la fonction : git status Fig.6
Le résultat montre que notre fichier a été modifié sans connaître les détails. Pour voir les lignes modifiées, tapons la fonction suivante :

```
$ git diff
```

pour obtenir le résultat suivant : Fig.7. Il est facile de voir les lignes modifiées, car elles sont identifiées par un signe '-' en début de ligne, si la ligne a été supprimée et d'un signe '+' au début, qui indique que la ligne a été ajoutée. Cette fonctionnalité permet de comparer entre 2 versions de commits. Maintenant, vous devez mettre à jour la branche Git, à partir de la fonction suivante :

```
$ git commit -am "exemple de mofication"
```

pour obtenir le résultat suivant : Fig.8. Et nous en profitons pour vérifier le journal des logs avec la fonction suivante : git log. Le nouveau résultat : Fig.9

Aller plus loin avec Git

Un des intérêts du versionning est de pouvoir récupérer du code ancien. Cette partie peut sembler inutile, mais par expérience, vous serez heureux de pouvoir le faire. Les cas flagrants sont par exemple : si du jour au lendemain, un nouveau message d'erreur apparaît alors que cette partie fonctionnait parfaitement. Si dans notre fichier index, vous avez perdu la ligne « hello world » et qu'il vous reste ceci

```
<?PHP
echo "(c) Programmez 2014<br />";
?>
```

Vous pouvez restaurer le contenu de ce fichier de la manière suivante :

Fig.4

```
hello@hello-1215B-1215B: ~/www/gitphp
hello@hello-1215B-1215B:~/www/gitphp$ git status
# On branch master
nothing to commit (working directory clean)
hello@hello-1215B-1215B:~/www/gitphp$
```

Fig.5

```
hello@hello-1215B-1215B: ~/www/gitphp
hello@hello-1215B-1215B:~/www/gitphp$ git log
commit 6ff933790741c921c1bcd7041b5b4c3295e15113
Author: hellos[illegible] <hello[illegible]@gmail.com>
Date:   Wed Jul 30 08:35:11 2014 +0200

    premier commit
hello@hello-1215B-1215B:~/www/gitphp$
```

Fig.6

```
hello@hello-1215B-1215B: ~/www/gitphp
hello@hello-1215B-1215B:~/www/gitphp$ git status
# On branch master
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working directory)
#
#       modified:   index.php
#
no changes added to commit (use "git add" and/or "git commit -a")
hello@hello-1215B-1215B:~/www/gitphp$
```

Tout d'abord, vous choisissez le numéro du dépôt et ensuite vous restaurez votre fichier :

```
$ git checkout f53094eb3d15d98d0d553838c0fdac6a0c2b591f
$ git checkout index.php -m 'recuperation'
```

Le résultat montre qu'un fichier a récupéré et vous retrouvez le code source d'origine :

```
<?php
echo "Hello World<br />";
echo "(c) Programmez 2014<br />";
?>
```

Rejoindre un projet Git

Un jour, vous pouvez avoir envie de contribuer et de participer à un projet PHP et celui qui vous intéresse propose Git comme logiciel. Pour rejoindre l'équipe, vous devez récupérer un dépôt existant. Suivant l'emplacement, vous devez exécuter une des lignes suivantes :

```
$ git clone http://siteInternet.com/nomDuDepot votreProjet
$ git clone votreLogini:votrePassword /nomDuDepot votreProjet
```

Après le téléchargement de l'ensemble du projet dans votre environnement de travail, vous pouvez commencer votre contribution. Par ailleurs, il est important de penser à commiter votre code quand celui-ci est réalisé. Pour cela, l'opération d'ajout et de modification est identique à ce qui a été vu plus haut.

Conclusion

Git est outil puissant et vous n'êtes pas obligé d'utiliser les lignes de commandes,car des interfaces existent.

Pour aller plus loin : http://git-scm.com/ - http://git-scm.com/doc - http://github.com/

Christophe **Villeneuve**

Fig.7

```
hello@hello-1215B-1215B: ~/www/gitphp
hello@hello-1215B-1215B:~/www/gitphp$ git diff
diff --git a/index.php b/index.php
index 3856f1f..5b54eb4 100644
--- a/index.php
+++ b/index.php
@@ -1,5 +1,4 @@
 <?php
-
 echo "Hello World<br />";
-
+echo "(c) Programmez 2014<br />";
 ?>
hello@hello-1215B-1215B:~/www/gitphp$
```

Fig.8

```
hello@hello-1215B-1215B: ~/www/gitphp
hello@hello-1215B-1215B:~/www/gitphp$ git commit -am "exemple de mofication"
[master f53094e] exemple de mofication
 1 file changed, 1 insertion(+), 2 deletions(-)
hello@hello-1215B-1215B:~/www/gitphp$
```

Fig.9

```
hello@hello-1215B-1215B: ~/www/gitphp
hello@hello-1215B-1215B:~/www/gitphp$ git log
commit f53094eb3d15d98d0d553838c0fdac6a0c2b591f
Author: hellos[illegible] <hello[illegible]@gmail.com>
Date:   Wed Jul 30 13:13:28 2014 +0200

    exemple de mofication

commit 6ff933790741c921c1bcd7041b5b4c3295e15113
Author: hellos[illegible] <hello[illegible]@gmail.com>
Date:   Wed Jul 30 08:35:11 2014 +0200

    premier commit
hello@hello-1215B-1215B:~/www/gitphp$
```

Fig.10

```
hello@hello-1215B-1215B: ~/www/gitphp
hello@hello-1215B-1215B:~/www/gitphp$ git chekout f53094eb3d15d98d0d553838c0fdac
6a0c2b591f
git: 'chekout' is not a git command. See 'git --help'.

Did you mean this?
        checkout
hello@hello-1215B-1215B:~/www/gitphp$ git commit index.php -m 'recuperation'
[master 7f4576a] recuperation
 1 file changed, 1 insertion(+), 2 deletions(-)
hello@hello-1215B-1215B:~/www/gitphp$
```

Assetic & Symfony2 : améliorer les performances de votre site

La gestion des assets fait partie intégrante du travail d'un développeur web. Ces fichiers (images, scripts clients, feuilles de styles, etc.) ont un impact direct sur les performances d'un site et notamment sur le temps de chargement de ses pages. Le framework Symfony2 met à la disposition du développeur divers outils afin de rendre cette tâche plus aisée. Voyons comment utiliser pleinement ces derniers afin d'améliorer les performances de votre site.

Emplacement et déploiement des assets

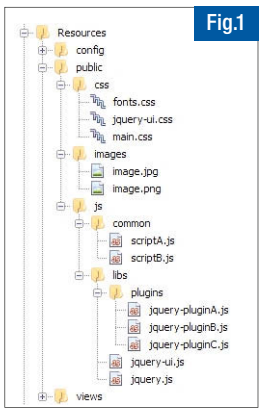
Dans Symfony2, les assets doivent être stockés dans le répertoire « **Resources/public** » du bundle auxquels ils sont liés. Ce répertoire n'étant pas accessible depuis l'extérieur, une manipulation est nécessaire afin de les « publier », c'est-à-dire de les copier dans un répertoire accessible publiquement. Cette opération est réalisée à l'aide de la commande :

```
php app/console assets:install
```

Cette commande va copier les assets de chaque bundle (du projet mais également des bundles tiers) vers le dossier « **web** » (par défaut, le seul répertoire accessible publiquement) en suivant la nomenclature suivante « (**web/**)**bundles/***nombundle* ».

Afin d'être prise en compte, toute modification, ajout ou suppression de fichier d'asset doit donc être suivi du redéploiement complet des assets. Afin d'éviter cette manipulation fastidieuse lors de la phase de développement, on peut utiliser l'option **–symlink** de la commande précédente. Au lieu d'effectuer une copie physique des fichiers (« hard mode »), celle-ci va créer un lien symbolique vers les dossiers « public » de chaque bundle, permettant ainsi la prise en compte immédiate des modifications. Attention en revanche à ne pas utiliser cette option en environnement de production pour des raisons de performance.

Fig.1



Faire référence aux assets

Pour l'exemple, considérons que l'on dispose d'un bundle AcmeDemoBundle qui possède les assets suivants : Fig.1.

Le layout TWIG devrait donc ressembler à ceci : Fig.2.

On peut remarquer que l'on fait référence au chemin des assets via la fonction « **asset()** ».

Cette dernière reçoit un chemin public relatif, c'est-à-dire celui qui correspond au chemin de l'asset une fois publié, et se charge de la traduire en url. Le code HTML d'une page ressemble alors à ceci : Fig.3.

Fig.2

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8" />
        <title>{% block title %}{% endblock %}</title>
        {% block stylesheets %}
            <link href="{{ asset('bundles/acmedemo/css/jquery-ui.css') }}" rel="stylesheet" />
            <link href="{{ asset('bundles/acmedemo/css/fonts.css') }}" rel="stylesheet" />
            <link href="{{ asset('bundles/acmedemo/css/main.css') }}" rel="stylesheet" />
        {% endblock %}
    </head>
    <body>
        {% block body %}{% endblock %}
        {% block javascripts %}
            <script src="{{ asset('bundles/acmedemo/js/libs/jquery.js') }}"></script>
            <script src="{{ asset('bundles/acmedemo/js/libs/jquery-ui.js') }}"></script>
            <script src="{{ asset('bundles/acmedemo/js/libs/plugins/jquery-pluginA.js') }}"></script>
            <script src="{{ asset('bundles/acmedemo/js/libs/plugins/jquery-pluginB.js') }}"></script>
            <script src="{{ asset('bundles/acmedemo/js/libs/plugins/jquery-pluginC.js') }}"></script>
            <script src="{{ asset('bundles/acmedemo/js/common/scriptA.js') }}"></script>
            <script src="{{ asset('bundles/acmedemo/js/common/scriptB.js') }}"></script>
        {% endblock %}
    </body>
</html>
```

Eviter les problèmes de cache

L'un des avantages de l'utilisation de la fonction « asset » est la possibilité de suffixer très facilement l'ensemble des assets avec un numéro de révision. Cette méthode, appelée **cache busting**, permet d'empêcher l'utilisation du cache navigateur après une montée en version des sources. Pour ceci, il nous suffit d'activer la fonction dans la configuration du framework :

app/config/config.yml

```
framework:
    # ...
    templating:
        # ...
        assets_version: v3
```

En saisissant ici, par exemple, la valeur « v3 » pour l'attribut « asset_version », chaque référence au chemin d'un asset ressemble désormais à cela :

```
<script src="/bundles/acmedemo/js/common/scriptB.js?v3"></script>
```

Il nous suffit donc d'incrémenter (manuellement ou via un script de mise en production) ce numéro de version à chaque montée de version afin d'éviter les problèmes récurrents de cache chez les utilisateurs que chaque développeur ne connaît que trop bien !

Aller plus loin grâce à Assetic

Nous venons de voir que Symfony propose de base des fonctions simples pour les assets.

Pour aller plus loin, le framework intègre Assetic, une solution tierce qui va permettre de pousser la gestion des assets en automatisant certains traitements.

Commençons par optimiser le chargement des fichiers JS : notre projet en utilise 7 que nous chargeons pour le moment individuellement. L'un des premiers services que peut rendre Assetic est d'automatiser le char-

Fig.3

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8" />
        <title>Titre de ma page</title>
        <link href="/bundles/acmedemo/css/jquery-ui.css" rel="stylesheet" />
        <link href="/bundles/acmedemo/css/fonts.css" rel="stylesheet" />
        <link href="/bundles/acmedemo/css/main.css" rel="stylesheet" />
    </head>
    <body>
        [...]
        <script src="/bundles/acmedemo/js/libs/jquery.js"></script>
        <script src="/bundles/acmedemo/js/libs/jquery-ui.js"></script>
        <script src="/bundles/acmedemo/js/libs/plugins/jquery-pluginA.js"></script>
        <script src="/bundles/acmedemo/js/libs/plugins/jquery-pluginB.js"></script>
        <script src="/bundles/acmedemo/js/libs/plugins/jquery-pluginC.js"></script>
        <script src="/bundles/acmedemo/js/common/scriptA.js"></script>
        <script src="/bundles/acmedemo/js/common/scriptB.js"></script>
    </body>
</html>
```

gement de ces derniers. Modifions notre layout comme ceci :

```
<!DOCTYPE html>
<html>
    ...
    <body>
        {% block body %}{% endblock %}
        {% block javascripts %}
            {% javascripts
                'bundles/acmedemo/js/libs/*.js'
                'bundles/acmedemo/js/libs/plugins/*.js'
                'bundles/acmedemo/js/common/*.js'
            %}
                <script src="{{ asset_url }}"></script>
            {% endjavascripts %}
        {% endblock %}
    </body>
</html>
```

On remarque l'utilisation du tag « javascripts » qui reçoit un ou plusieurs chemins en paramètre ainsi que le template du code HTML qui va être utilisé pour les charger (où « asset_url » sera remplacé automatiquement par l'url du fichier).

Ces chemins peuvent accepter des jokers (« * ») permettant ainsi le chargement automatique du contenu d'un répertoire (mais pas de ses sous-répertoires qui doivent être chargés indépendamment). Il est à noter que, dans ce cas, les fichiers sont chargés dans l'ordre alphabétique. Il peut être parfois intéressant de contrôler l'ordre de chargement des fichiers en les préfixant par exemple d'un numéro de priorité (ex : 00-jquery.js).

Regardons maintenant le rendu HTML suite à la modification du layout :

```
<html>
    ...
    <body>
        ...
            <script src="/js/04202e0_part_1_jquery-ui_1.js"></script>
            <script src="/js/04202e0_part_1_jquery_2.js"></script>
            <script src="/js/04202e0_part_2_jquery-pluginA_1.js"></script>
```

On remarque que les fichiers sont désormais préfixés par un hash et sont appelés depuis un répertoire « **js** » qui n'existe pas physiquement à l'heure actuelle.

Lorsque l'on se trouve en environnement de développement, Symfony va intercepter ces appels et les rediriger vers le chemin public du fichier adéquat. Ce comportement est appréciable car il permet de debugger facilement d'éventuelles erreurs JS en pouvant identifier le fichier d'où provient l'erreur. En production, en revanche, Assetic va nous rendre un second service en concaténant automatiquement l'ensemble des fichiers en un unique asset réduisant ainsi drastiquement le nombre de requêtes, et en améliorant du même coup le temps de chargement de nos pages.

Pour cela, de la même façon qu'il faut publier préalablement les assets, il faut également demander à Assetic de réaliser la concaténation des fichiers et d'en publier le fichier résultant.

On réalise cette opération via la commande suivante :

```
php app/console assetic:dump --env=prod
```

Assetic génère alors des fichiers dans le répertoire **web/js**.

On peut vérifier que le rendu HTML est maintenant différent en environnement de production :

```
<html>
    ...
    <body>
        ...
            <script src="/js/1af6030.js?v1"></script> <!-- Contient le code des 7 fichiers JS -->
```

Il est à noter que ces manipulations devront être renouvelées à chaque mise en production afin que les modifications soient prises en compte.

Petite astuce:

Si vous chargez vos fichiers avec des chemins contenant des jokers (« * »), il peut arriver qu'au cours de vos développements, l'ajout (ou la suppression) d'un fichier ne soit pas pris en compte. Pour forcer Assetic à rechercher les nouveaux fichiers, modifiez une ligne de paramètres du tag « javascripts » (en remplaçant par exemple « *.js » par « *.* » ou en inter-changeant l'ordre de 2 lignes), rafraîchissez votre page puis annulez vos modifications; cela forcera Assetic à les prendre en compte.

L'intérêt d'Assetic : les filtres

L'un des intérêts principaux d'Assetic est qu'il permet de piloter des outils externes afin qu'ils effectuent des tâches sur nos assets. De base Assetic sait piloter une trentaine de filtres qui vont permettre par exemple de compresser, minifier ou encore compiler nos scripts. Les outils pilotés par Assetic doivent être installés sur la machine où le déploiement a lieu.

Continuons donc à optimiser notre projet : Actuellement, Assetic a concaténé nos fichiers JS en un unique fichier permettant de gagner 6 requêtes sur chacune de nos pages mais la taille de ce fichier n'a pas été optimisée. Nous allons donc demander à Assetic de minifier les assets en utilisant l'utilitaire UglifyJS.

Nous ne détaillerons pas l'installation d'UglifyJS, celle-ci se réalise via NPM (Gestionnaire de paquets de Node.js) et de nombreux tutoriaux traitant de ce sujet sont disponibles sur le web.

On indique tout d'abord à Assetic que l'on souhaite piloter UglifyJS via le fichier de configuration (adaptez le chemin des binaires selon votre configuration) :

app/config/config.yml

```
# Assetic Configuration
assetic:
    debug: "%kernel.debug%"
    filters:
        cssrewrite: ~
        uglifyjs2:
            bin: /opt/node/bin/uglifyjs
            node: /opt/node/bin/node
            node_paths: [/opt/node/lib/node_modules/]
```

On demande ensuite à Assetic d'appliquer ce filtre sur nos assets via une directive du tag « javascripts ».

```
{% javascripts filter="?uglifyjs2"
    'bundles/acmedemo/js/libs/*.js'
    'bundles/acmedemo/js/libs/plugins/*.js'
    'bundles/acmedemo/js/common/*.js'
%}
```

On remarque que le nom du filtre est préfixé d'un « ? ». Cela permet de n'appliquer ce filtre qu'en environnement de production. En effet la minification n'apporte aucun bénéfice en développement (il devient plus difficile de débugger) et entraîne une baisse sensible des performances (l'opération de minification devrait être effectuée à chaque chargement de l'asset). Si l'on « dump » les assets une nouvelle fois, nos 7 fichiers sont désormais concaténés en un seul et unique fichier qui a été minifié, améliorant une nouvelle fois les performances de chargement de nos pages.

Optimisation des CSS

Attaquons-nous maintenant aux fichiers CSS de notre projet.

Ils sont au nombre de 3, nous allons donc commencer par les charger automatiquement. Pour cela nous modifions le layout TWIG comme ceci :

```
<head>
    ...
    {% block stylesheets %}
        {% stylesheets filter='cssrewrite'
            'bundles/acmedemo/css/*.css'
        %}
            <link href="{{ asset_url }}" rel="stylesheet" />
        {% endstylesheets %}
    {% endblock %}
</head>
```

Cette fois-ci nous utilisons le tag « stylesheets » qui est l'équivalent du tag « javascripts » pour les feuilles de style CSS. On remarque également

l'utilisation d'un filtre « cssrewrite ». En effet, si l'on regarde le rendu HTML, on remarque que les fichiers CSS sont désormais appelés depuis le chemin « css ».

```
<head>
    ...
    <link href="/bundles/acmedemo/css/jquery-ui.css?v1" rel="stylesheet" />
    <link href="/bundles/acmedemo/css/fonts.css?v1" rel="stylesheet" />
    <link href="/bundles/acmedemo/css/main.css?v1" rel="stylesheet" />
    ...
```

L'utilisation de ce chemin, différent de l'emplacement réel du fichier, peut casser les liens vers d'éventuelles images ou polices de caractères contenus dans le CSS. Le filtre cssrewrite va automatiquement parcourir les fichiers CSS et réécrire à la volée les chemins qui y sont contenus afin qu'ils fonctionnent avec ce nouveau chemin « /css ».

De la même façon que pour les fichiers javascripts, les fichiers seront chargés individuellement en environnement de développement (ou tout environnement ou le mode de debug est activé) mais concaténés automatiquement en production.

Ainsi si l'on « dump » de nouveau les assets et que l'on se place en environnement de production, on obtient un seul appel vers les CSS dans le code généré :

```
<link href="/css/e1ff442.css?v1" rel="stylesheet" />
```

Tout comme pour les fichiers JS, d'autres filtres existent également et permettent, entre autres, de minifier les feuilles de styles (avec l'aide du filtre UglifyCSS par exemple).

Dernière étape : Les images

Nous avons donc amélioré les performances de notre projet sur les appels JS et CSS, reste maintenant à améliorer nos images. Assetic permet également de s'attaquer à ce point grâce à de nombreux filtres spécialisés dans les formats web les plus courants.

Un appel vers une image se traduit normalement par le code suivant :

```
<img src="{{ asset('bundles/acmedemo/images/image.png') }}" />
```

Afin qu'Assetic prenne en charge nos images nous allons le remplacer par le code suivant :

```
{% image 'bundles/acmedemo/images/image.png' %}
    <img src="{{ asset_url }}" />
{% endimage %}
```

Nous remarquons l'utilisation du tag « image », semblable aux 2 précédents, qui accepte en paramètre le chemin de l'image et le template HTML à utiliser.

Nous allons ensuite utiliser les filtres **OptiPNG** et **JpegOptim** afin d'optimiser les images PNG et JPEG de notre projet (nous considérons que les 2 utilitaires du même nom sont installés sur la machine).

Pour cela nous configurons Assetic comme ceci :

app/config/config.yml

```
# Assetic Configuration
assetic:
    # ...
    filters:
        # ...
        optipng:
            bin: /usr/bin/optipng
            level: 3
            apply_to: "\.png$"
        jpegoptim:
            bin: /usr/bin/jpegoptim
            strip_all: true
            max: 50
            apply_to: "\.jpe?g$"
```

Nous déclarons donc les filtres « optipng » et « jpegoptim » et pour chacun d'eux les chemins des exécutables et les options d'optimisation. Ici, on souhaite par exemple limiter la qualité à 50% pour les fichiers JPEG et à 3 pour les fichiers PNG ainsi que supprimer l'ensemble des données EXIM présentes dans les fichiers JPEG. On applique ensuite automatiquement ces filtres sur les fichiers qui portent une extension en PNG (pour OptiPNG) et JPG/JPEG (pour JpegOptim) grâce à la directive « apply_to ». Il n'est donc pas utile de spécifier le(s) filtre(s) à utiliser dans les vues, ce fichier de configuration pilote l'ensemble des images du projet qui utilisent le tag « image ».

Le traitement des images nécessitant du temps et des ressources il est préférable de l'activer uniquement en environnement de production (en activant les filtres uniquement dans config_prod.yml par exemple). Les images seront alors traitées lors de la mise en production via la commande de « dump » d'Assetic.

Conclusion

Assetic permet en peu de temps d'améliorer à la fois notre confort de développement (via le chargement automatique des assets par exemple) mais également les performances finales de notre site en limitant le nombre et la taille des requêtes de nos pages. Mais les possibilités offertes par cet outil ne s'arrêtent pas là ! Il peut par exemple simplifier les développements et déploiements en se chargeant de compiler des feuilles SASS/LESS ou encore des scripts CoffeeScript ou Dart !

Bien que son intégration au sein de Symfony en fasse un outil à la fois simple et puissant à utiliser, il peut également être utilisé en tant que librairie sur un projet n'utilisant pas ce framework.

N'hésitez pas à vous rendre sur le site du projet qui recense les filtres disponibles et dont la liste évolue régulièrement.

Pour aller plus loin

Assetic : https://github.com/kriswallsmith/assetic

Antoine **PACAUD**
Architecte technique chez Webnet www.webnet.fr

Optimiser vos développements PHP : notre sélection d'outils !

Nous allons voir qu'il est important d'être réactif. Une application doit afficher le contenu le plus rapidement possible, tout en contrôlant les ressources. Mais il faut aussi valider le code pour gagner en clarté et assurer la gestion du cycle de vie de l'application. Nous vous proposons une sélection d'outils.

Améliorer les performances

L'amélioration des performances a toujours été une problématique pour le développeur. Les améliorations touchent le matériel, le serveur, le réseau, la gestion des fichiers, etc. Cependant, la performance d'une application dépend beaucoup de l'analyse et du profiling. Ces techniques permettent de détecter les codes, les modules lents.

xDebug

Xdebug est un outil qui existe depuis de nombreuses années. Il est très utilisé pour déboguer les applications PHP et aider les développeurs à trouver les anomalies. Cependant, il est possible d'activé le mode *profiler* pour effectuer une simple analyse PHP.

Cette extension existe depuis le début du projet, mais souvent mal connu des programmeurs. Si vous souhaitez utiliser la fonction de profilage, vous devez tout d'abord installer l'extension et de l'activer dans php.ini

```
xdebug.profiler_enable = On
xdebug.profiler_output_dir="/votreDossier"
```

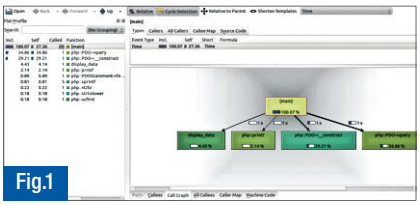


Fig.1

Le résultat que vous obtiendrez sera stocké dans le dossier que vous aurez configuré. Les informations concernent les fonctions et méthodes lentes de votre projet que vous pourrez visualiser avec un outil comme Webgrind ou cachegrind **Fig.1.** Site : http://xdebug.org

xhprof

Xhprof est une extension PECL pour PHP, créé par Facebook et permet d'analyser votre application PHP sous la forme d'un profiler hiérarchique. Le résultat se rapproche de xDebug, qui s'appuie sur le même principe de callgraph et donne des statistiques plus précises sur le matériel. Bien entendu, le résultat propose une liste de tous les appels de fonctions, leur temps de consommation au niveau de la mémoire et du CPU. L'installation est simple puisqu'il s'agit d'une extension PHP. Au niveau de la configuration, celle-ci passe par l'ajout des lignes suivantes dans php.ini :

```
extension= xhprof.so
xhprof.output_dir = /votreDossier
```

Fig.2.

Enfin, vous pouvez l'installer directement sur votre serveur de production. **Fig.3.** Site : https://github.com/facebook/xhprof

Qafoo pour PHP

Qafoo est un autre outil de profiling et d'analyses de performances. Son but est de collecter les temps de réponse de PHP et des données de profilage détaillées. Le rendu est immédiat pour vous aider à améliorer les performances de votre application. Très utile si vous développez sur des CMS ou des plateformes de e-commerce. L'outil s'est principalement intéressé sur la manière de transmettre les données à partir d'une requête PHP pour identifier les points de surcharge. Il a obtenu de très bons résultats en utilisant les ports UDP et les sockets pour la transmission des données. **Fig.4.** Le résultat affiche un rendu sur le nombre de requêtes exécutées, le temps de chacune d'elles, le nombre de connexions ouvertes, le nombre d'objets ORM et l'ensemble des informations utiles. Un rapport détaillé vous aide à obtenir une vue d'ensemble des parties lentes et rapides de votre code. **Fig.5.** Enfin le système est ouvert et vous pourrez ajouter vos propres bibliothèques. Site : http://www.qafoo.com

SensioLabsProfiler

SensioLabsProfiler est un nouvel outil (en bêta privée à la rédaction de cet article) qui simplifie la mesure des performances des applications PHP. Une des principales innovations est une interface de visualisation permettant de naviguer dans le graphe d'appel comme dans une carte : zoom, glisser-

Fig.2

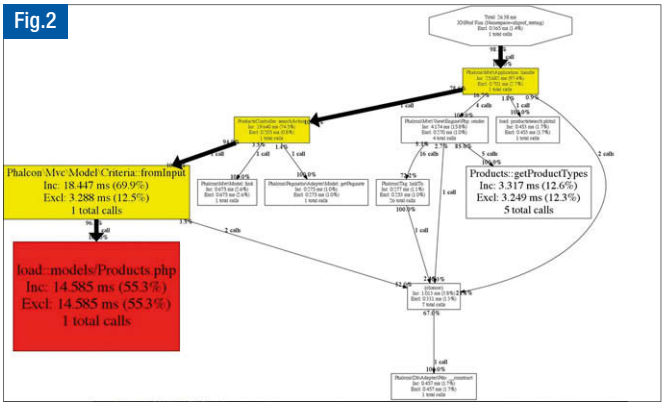
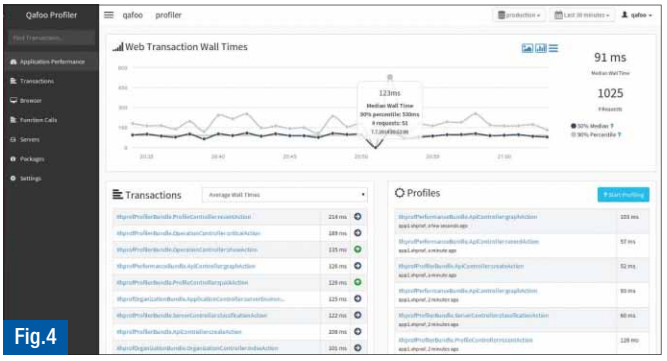


Fig.3

Displaying top 100 functions: Sorted by Excl. Wall Time (microsec) [display all]

Function Name	Calls	Calls%	Incl. Wall Time (microsec)	IWall%	Excl. Wall Time (microsec)	EWall%	Incl. CPU (microsecs)	ICpu%	Excl. CPU (microsec)	ECPU%	Incl. MemUse (bytes)
load::models/Products.php	1	0.8%	14,585	55.3%	14,585	55.3%	346	5.4%	346	5.4%	43,528
Phalcon\Mvc\Model\Criteria::fromInput	1	0.8%	18,447	69.9%	3,288	12.5%	1,515	23.7%	831	13.0%	87,544
Products::getProductTypes	5	3.8%	3,317	12.6%	3,249	12.3%	1,029	16.1%	958	15.0%	57,992
Phalcon\Mvc\Application::handle	1	0.8%	25,682	97.4%	701	2.7%	5,981	93.4%	683	10.7%	355,296
Phalcon\Mvc\Model::find	1	0.8%	675	2.6%	675	2.6%	457	7.1%	457	7.1%	16,288
Phalcon\Db\Adapter\Pdo::__construct	1	0.8%	457	1.7%	457	1.7%	217	3.4%	217	3.4%	8,664
load::products/search.phtml	1	0.8%	453	1.7%	453	1.7%	288	4.5%	288	4.5%	26,144
main()	1	0.8%	26,380	100.0%	365	1.4%	6,401	100.0%	128	2.0%	402,568
{closure}	7	5.4%	1,013	3.8%	331	1.3%	777	12.1%	308	4.8%	99,544
Phalcon\Paginator\Adapter\Model::getPaginate	1	0.8%	275	1.0%	275	1.0%	276	4.3%	276	4.3%	24,928
Phalcon\Mvc\View\Engine\Php::render	4	3.1%	4,174	15.8%	270	1.0%	1,864	29.1%	231	3.6%	99,664
Phalcon\Tag::linkTo	26	20.0%	277	1.1%	253	1.0%	292	4.6%	268	4.2%	9,024
ProductsController::searchAction	1	0.8%	19,640	74.5%	205	0.8%	2,479	38.7%	184	2.9%	142,392
Phalcon\Loader::autoLoad	5	3.8%	174	0.7%	128	0.5%	122	1.9%	89	1.4%	14,032
Phalcon\Config\Adapter\Ini::__construct	1	0.8%	122	0.5%	122	0.5%	123	1.9%	123	1.9%	4,480
Phalcon\Session\Adapter::start	1	0.8%	115	0.4%	115	0.4%	117	1.8%	117	1.8%	34,328
Phalcon\DI::set	8	6.2%	80	0.3%	80	0.3%	65	1.0%	65	1.0%	2,592
Elements::getMenu	1	0.8%	107	0.4%	59	0.2%	108	1.7%	52	0.8%	5,632
Security::beforeDispatch	1	0.8%	243	0.9%	52	0.2%	244	3.8%	46	0.7%	47,208
Phalcon\Loader::__construct	1	0.8%	51	0.2%	51	0.2%	15	0.2%	15	0.2%	1,160
Elements::getTabs	1	0.8%	96	0.4%	51	0.2%	97	1.5%	45	0.7%	4,504
Phalcon\Di\FactoryDefault::__construct	1	0.8%	47	0.2%	47	0.2%	49	0.8%	49	0.8%	17,736

Fig.4



déplacer, clique sur une fonction pour afficher les informations détaillées, etc. L'outil est très fluide, même avec des applications complexes comme *Drupal*. Pour faciliter l'identification des nœuds problématiques, les appels les plus coûteux sont mis en avant. Fig.6.

Une fonctionnalité très pratique de cette interface est la comparaison de graphes : imaginez que vous ayez identifié un bottleneck dans votre autoloader. L'outil permet de vérifier immédiatement que votre patch améliore effectivement la performance : un clic dans le navigateur pour prendre la mesure avant le patch, un autre après le patch et on obtient un graph comparatif, avec en bleu les appels ayant « refroidis », et en rouge ceux ayant « chauffés ». Fig.7.

SensioLabsProfiler est basé sur plusieurs composants qui facilitent et sécurisent la collecte et la visualisation des profils de performances. Le cœur du système est une sonde à déployer dans le moteur PHP, via une extension. Dérivée de l'outil open source *uprofiler* (lui-même dérivé de *xhprof*), c'est cette sonde qui collecte les temps d'exécution, la consommation mémoire de chaque appel de fonction PHP, etc. Elle envoie les données collectées à un agent via un *socket* local. L'agent a pour rôle d'agréger les mesures puis de les envoyer à un serveur hébergé où se trouve l'interface de visualisation. Une fois que l'extension et l'agent sont installés sur un poste de travail, le développeur utilise son navigateur pour piloter toute la chaîne ! À noter que pour une meilleure qualité de mesure, l'outil va exécuter plusieurs fois votre page pour obtenir une moyenne stable et représentative.

À terme, *SensioLabsProfiler* pourra être utilisé sur des serveurs d'intégration et/ou de production : grâce à un système d'activation basé sur des signatures par clés publiques/privés, la sonde (l'extension PHP) pourra être activée à la demande par les développeurs, tout en ayant un impact zéro pour les utilisateurs.

Site : http://profiler.sensiolabs.com/

Analyse statique

La notion d'analyse statique a toujours existé. Il s'agit d'une variété de méthodes pour obtenir des informations sur le comportement d'une application lors de son exécution. Il ne faut pas confondre avec l'analyse dynamique que l'on trouve dans les outils de débug et de profiling.

Slow Calls | Significant Calls | Counters 10 | Custom Measurements 8

Function Name	Count		Incl. Wall Time	
Symfony\Component\EventDispatcher\EventDispatcher::dispatch	6	0.0%	97.8 ms	83.7%
Symfony\Component\DependencyInjection\Container::get	124	0.6%	65.2 ms	55.7%
Symfony\Component\Security\Http\Firewall::onKernelRequest	2	0.0%	50.3 ms	43.0%
Composer\Autoload\ClassLoader::findFile	552	2.6%	25.8 ms	22.1%
Twig_Template::render	1	0.0%	19.8 ms	16.9%
Symfony\Component\HttpKernel\Fragment\FragmentHandler::render	1	0.0%	12.0 ms	10.2%
Doctrine\ORM\Internal\Hydration\AbstractHydrator::hydrateAll	4	0.0%	5.2 ms	4.4%
Doctrine\ORM\EntityManager::flush	1	0.0%	4.7 ms	4.0%
Symfony\Component\HttpKernel\Kernel::boot	1	0.0%	4.4 ms	3.7%
Doctrine\DBAL\Connection::executeQuery	5	0.0%	3.9 ms	3.3%

Show More

Fig.5

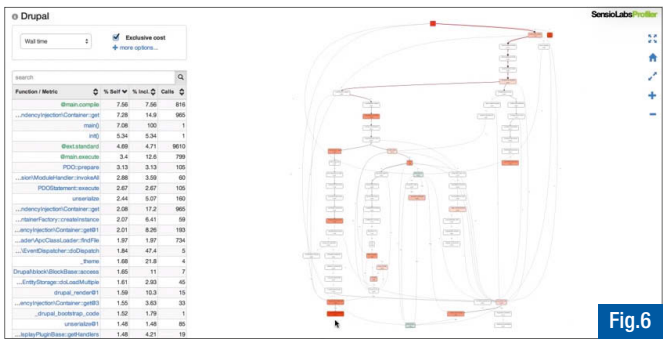


Fig.6

SensioLabsInsight

SensioLabsInsight permet d'analyser automatiquement le code et le comportement d'une application et d'en détecter les défauts en utilisant +100 points de contrôles afin de s'assurer qu'elle est sécurisée, fiable, maintenable et qu'elle suit les bonnes pratiques et standards.

La liste des projets d'un dépôt peut être récupérée via SSO sur GitHub et Bitbucket. Il suffit alors de sélectionner le projet souhaité pour lancer l'analyse. Un projet privé se configure simplement avec l'URL du dépôt et une clé SSH fournie. La configuration des règles peut s'optimiser en sélectionnant un type de projet parmi une liste prédéfinie (Librairie PHP, Projet Web PHP, Symfony2/symfony1, Silex, Laravel, Bundle Symfony2, Plugin symfony1, module Drupal...).

Les résultats d'analyses contextuelles, statiques ou dynamiques, présentent tout leur intérêt via des indicateurs d'évolution dans le temps, et fournissent une indication de la dette technique.

Les violations et alertes sont remontées sur des règles de Sécurité, Risques de bugs, Performance, Architecture, Code mort, Lisibilité et Style de code, et sont assorties d'un niveau de criticité.

Les analyses peuvent être partagées entre collaborateurs ayant un compte SensioLabsConnect, ou encore avec un manager via un tableau de bord spécialement conçu. Chaque itération permet de gagner des médailles reflétant la qualité du code qui peuvent être publiées sous forme de widget pour affichage sur plateformes communautaires. Enfin, les analyses publiques permettent un suivi de tous les projets Open source référencés.

Un des avantages de l'outil réside dans l'accès à la documentation en ligne et de l'aide contextualisée, références et exemples permettant la résolution rapide des erreurs rencontrées.

SensioLabsInsight permet ainsi d'aller au-delà d'un audit qualité ponctuel réalisé par un consultant, grâce à un pilotage et à un suivi quotidien permettant de détecter efficacement les défauts. Ceux-ci sont détectés en amont, et le lancement automatique de l'analyse permet de limiter l'impact sur le processus de développement. Fig.8.

Site : http://insight.sensiolabs.com

Static code analysis tools

Il s'agit d'une initiative récente pour regrouper dans une suite logicielle plusieurs outils open source. Cette suite a pour but d'effectuer une analyse statique de votre code. On y trouve :

PHP Mess Director, PHP Code Sniffer, PHP Coding Standard Fixer, PHPUnit, PHP cope-paste detector, PHP lines-of-code, PHP documentor, PHP code browser.

Site : https://github.com/ManageWP/static-code-analysis-tools

PHP Mess Detector (PHPMD)

Il s'agit d'une application à installer avec PEAR. Il applique des règles de programmation pour vérifier la qualité de votre code.

La liste des règles est très variée. Ainsi, la détection se tourne vers une taille de code trop importante, un code lié aux problèmes de conceptions de

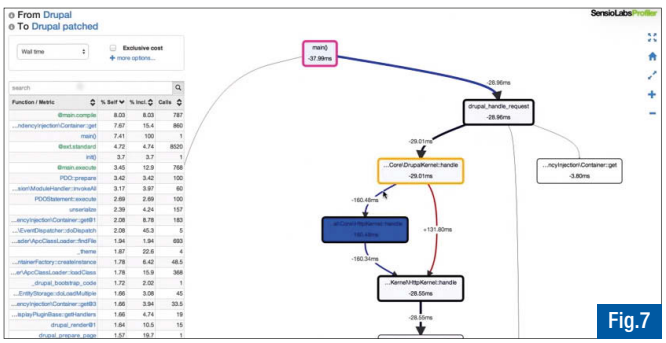


Fig.7

logiciel et design, les problèmes de nommages qui sont trop longs ou trop courts, de plus la détection de propriété non utilisée est repérée pour éviter de maintenir du code inutile. Il permet aussi de détecter des règles controversées comme la détection des SuperGlobals, des camelCases. La configuration passe par un fichier XML si vous ne souhaitez pas l'utiliser avec le Tools, de la manière suivante :

```
<target name="phpmd">
<phpmd file="${project.basedir}/${source}/www">
<formatter type="xml" outfile="${project.basedir}/build/phpmd.xml" />
</phpmd>
</target>
```

pour s'exécuter de la manière suivante :

```
$ cd /votreProjet
$ phpmd source html /chemin/phpmd.xml > /chemin/resultat/votre
resultat.html
```

Fig.9. Site : http://phpmd.org

PHP Code Sniffer

PHP Code Sniffer un autre moteur pouvant s'utiliser indépendamment du Tools. Son but est de repérer les violations des standards dans un code. En résumé, il effectue de nombreux contrôles de qualité, avec la possibilité de le configurer pour suivre le standard de Symfony2
Son utilisation s'effectue de la manière suivante :

```
$ phpcs /votreProjet/src/FunkyBundle/Controller/FunkyController.php
```

Fig.10.

PHP Coding Standard Fixer

Il s'agit d'un petit outil dont l'objectif est de fixer votre code pour respecter les standards de développements (les normes PSR-1 et PSR-2). Cela se traduit par transformer les tabulations avec un certain nombre d'espaces, comme dans l'exemple suivant qui va conformer vos différents fichiers selon le standard PSR2 :

```
$ PHP-cs-fixer fix /votreDossier -fixers=identification
```

Site : http://cs.sensiolabs.org/

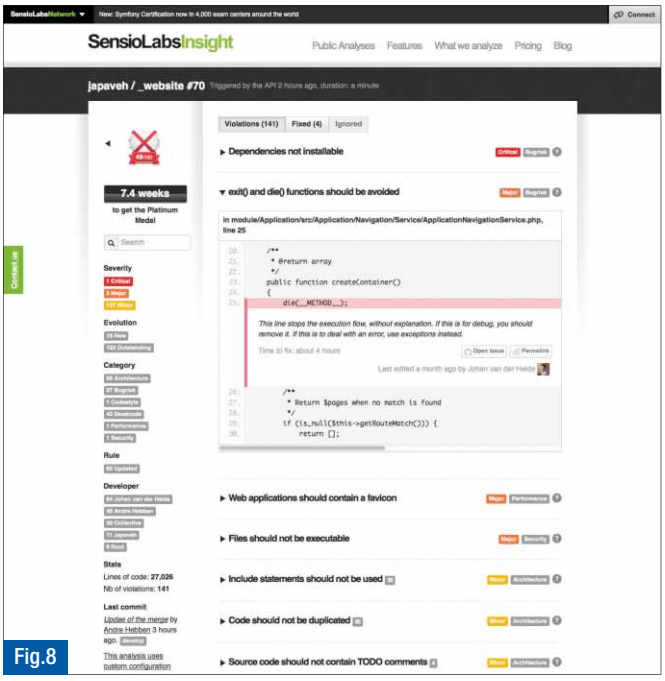


Fig.8

PHP copy-paste detector

Module très utile, car il propose des métriques sur la duplication de code : si votre code a été copié/collé dans différentes fonctions ou pages

PHP lines-of-code

PHP lines of code va rechercher le nombre de lignes de codes dans vos fonctions, le nombre de structures, les constantes, etc. Ainsi vous obtenez une mesure globale de votre projet, au niveau de la taille de celui-ci.

PHP code browser

PHP_CodeBrowser est un outil destiné à être utilisé avec CruiseControl. Il propose un navigateur de codes pour vos fichiers PHP avec la coloration syntaxique pour repérer facilement les zones d'erreurs. Ces zones seront codifiées par couleurs. Il est compatible avec les autres outils d'assurance qualité comme PHPUnit ou PHP_CodeSniffer. La visualisation des erreurs, des avertissements ou des avis, trouvé par type d'outils est le grand intérêt de cet outil. Site : svn://phpunit.de/phpunit/phpcb/trunk PHP_CodeBrowser

À vous de jouer !

Christophe **Villeneuve**
consultant IT pour Neuros, auteur du livre "Drupal7 en mode avancé" aux éditions Eyrolles et auteur éditions ENI, Rédacteur pour WebRIVER, membre des Teams DrupalFR, AFUP, LeMug.fr, Drupagora, PHPTV..

Nicolas **Grekas**
Conférencier AFUP, merger Symfony2, auteur de patchwork/utf8, product manager chez SensioLabs.

Fig.9

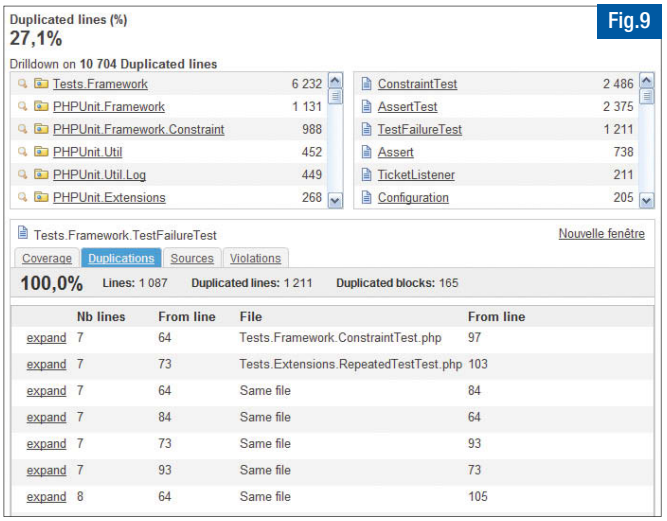
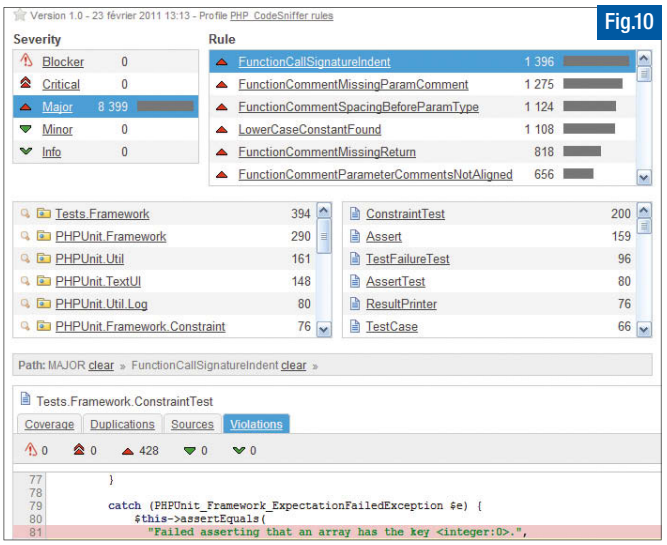


Fig.10



Introduction au développement AngularJS

Il est de plus en plus courant de développer des applications HTML/JavaScript destinées à être utilisées comme des applications de bureau classiques, et ce, pour bénéficier de tous les avantages proposés par une application distribuée au travers d'un navigateur.

On parle généralement de SPA (Single Page Application) pour qualifier ce type d'applications, qui se composent d'une seule et unique page au sein de laquelle l'utilisateur va pouvoir naviguer en chargeant les données et vues, de manière asynchrone, via des requêtes AJAX. AngularJS est un Framework gratuit et open-source, développé par Google. Il repose sur le principe de la programmation déclarative : enrichir le markup HTML et laisser le soin au Framework d'interpréter ces déclarations pour charger les vues et les données qui les composent au bon moment, en se basant sur des patterns simples tel que : MVC (Model-View-Controller), MVVM (Model-View-ViewModel) ou encore l'injection de dépendance. Dans cet article, nous vous présenterons les bases d'AngularJS qui vous permettront de comprendre rapidement la structure d'un projet mais aussi les différentes directives à utiliser dans une application pour afficher des données à l'utilisateur (binding) et lui permettre de naviguer d'une vue à l'autre.

Structure d'une application

Une application AngularJS est organisée autour de six éléments:

- Les vues, qui correspondent aux pages qui seront affichées à l'utilisateur.
- Les modèles, qui contiennent les données à afficher dans les vues.
- Les contrôleurs, dont le rôle est de gérer la liaison entre les modèles et les vues notamment en chargeant le modèle et en répondant aux actions de l'utilisateur sur la vue.
- Les services, chargés d'encapsuler une fonctionnalité métier ou technique et permettant de partager des données entre contrôleurs.
- Les filtres, utilisés pour formater une donnée du modèle avant d'être affichée dans la vue.
- Les directives, étendant la syntaxe HTML en rajoutant des balises ou attributs liés à un comportement et/ou à un template.

Les contrôleurs, services, filtres et directives sont déclarés dans des modules. Le rôle d'un module est de regrouper logiquement les éléments d'une application. Généralement, tous les éléments d'une application sont regroupés dans un seul et même module. Mais dans certains cas, il peut être pratique d'en utiliser plusieurs afin de délimiter les zones fonctionnelles de l'application (Ex : Facturation, Gestion des commandes, Support, etc...). Le modèle étant associé au contrôleur via la fonctionnalité d'« injection », et les vues étant des ressources globales à l'application, ils ne sont pas liés à un module. La création d'un module se fait de la manière suivante :

```
var myModule = angular.module('myModule', []);
```

Ou

```
var myModule = angular.module('myModule', ["ngRoute"]);
```

Le premier paramètre à renseigner correspond au nom du module, le deuxième est un tableau contenant la liste des modules dépendants.
Pour récupérer un module, il faut utiliser la syntaxe ci-dessous :

```
var myModule = angular.module('myModule');
```

Une fois le module créé et récupéré, nous pouvons y déclarer des contrôleurs de la manière suivante :

```
myModule.controller('controller', function($scope) {...});
```

Le premier paramètre est le nom de notre contrôleur, le deuxième est une fonction permettant de le créer. Le paramètre $scope, correspondant au modèle, est injecté par AngularJS. C'est ce paramètre $scope que nous allons enrichir afin de partager de la donnée entre le contrôleur et la vue.

La création de services, filtres et directives se fait de façon similaire à la création d'un controller :

```
myModule.service('myService', function() {...});
myModule.filter("myFilter", function() {...});
myModule.directive("myDirective", function() {...});
```

Par défaut, AngularJS intègre un certain nombre de services (Ex : $http permettant d'effectuer des requêtes HTTP), de filtres (Ex : currency permettant d'afficher et de formater un nombre sous format monétaire) et de directives (Ex : ng-show permettant d'afficher un élément HTML en fonction d'une valeur booléenne).

Injection de dépendances

Il est possible d'utiliser un service dans un filtre, dans une directive, dans un contrôleur ou dans un autre service. Pour cela, AngularJS utilise de l'injection de dépendances. Lors de la déclaration d'un élément dans un module, nous pouvons lui spécifier une liste de dépendances passés en paramètre de la fonction de construction.

```
myModule.controller('myController', function($scope, myService)
{...});
```

Nous indiquons ici à AngularJS qu'il faudra injecter le modèle, représenté par $scope, ainsi qu'un service appelé myService, et précédemment enregistré dans notre module. Lorsque AngularJS instanciera notre controller, il résoudra la dépendance en recherchant un élément portant le nom « myService » dans l'ensemble des modules chargés (allant du module courant, aux modules Core d'AngularJS, jusqu'aux modules dépendants).

Exemple d'une application AngularJS : MyTodoList.

Le but de cette application sera de proposer la possibilité de gérer un ensemble de tâches à réaliser (suppression, ajout et validation) pour un utilisateur donné. Nous allons donc créer l'arborescence de notre projet :

- Index.html
- Js
  - Controllers
    - MainController.js
  - Services
    - TasksService.js
  - App.js

Le fichier « racine » index.html est la page de base de notre projet. Le dossier le plus important de ce projet est le dossier Js, contenant l'ensemble de nos fichiers JavaScript. Sous ce dossier nous trouvons, un dossier Controllers contenant la déclaration de nos contrôleurs, un dossier Services contenant l'ensemble des services exposés dans notre application, et enfin, le fichier App.js qui est le point d'entrée de notre application. Notre fichier App.js ne va contenir pour le moment, qu'une seule et unique ligne de code :

```
var todoListApp = angular.module('todolist', []);
```

Nous créons donc notre module « todolist », qui ne dépend d'aucun module externe, puis nous allons ensuite créer notre service tasksService dans le fichier TasksService.js du dossier Services.

```
var todoListApp = angular.module('todolist');
todoListApp.service("tasksService", function () {
```

```
    var tasks = [];
    this.add = function (task) {
        tasks.push(task);
    };
    this.remove = function (index) {
        tasks.splice(index, 1);
    };
    this.done = function (task) {
        task.isDone = true;
    };
    this.getTasks = function () {
        return tasks;
    };
});
```

Ce service va nous permettre d'agir sur les tâches à réaliser par l'utilisateur. Il nous permet de réaliser un CRUD sur les tâches de l'utilisateur. Pour le moment ce service est très simple et ses tâches sont stockées directement dans un tableau d'objet (aucune persistance n'est réalisée).

```
var todoListApp = angular.module('todolist');
todoListApp.controller('mainController', function ($scope, tasks
Service) {
    $scope.tasks = tasksService.getTasks();
    $scope.taskTitle = "";
    $scope.addTask = function () {
        if (!$scope.taskTitle)
            return;
        var task = {
            isDone: false,
            title: $scope.taskTitle
        };
        tasksService.add(task);
        $scope.taskTitle = "";
    };
    $scope.removeTask = function (index) {
        tasksService.remove(index);
    };
    $scope.doneTask = function (task) {
        tasksService.done(task);
    };
});
```

Voici le cœur de notre application, le contrôleur main va nous permettre de réaliser l'ensemble des opérations nécessaires. Nous exposons à travers la variable $scope (qui, comme expliqué plus haut, est l'image de notre modèle coté vue) un ensemble de fonctions qui nous permettront de traiter diverses opérations sur les tâches à réaliser par l'utilisateur. Créons la vue de notre application à l'aide des directives fournies par AngularJS:

```
<html ng-app="todolist">
<head>
  <meta charset="utf-8" />
  <title>Todo List</title>
  <script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.2.
16/angular.
min.js"></script>
</head>
<body ng-controller="mainController">
  <span>Nouvelle tâche : </span><input type="text" ng-model="taskTitle">
<button ng-click="addTask()">+</button>
  <ul ng-repeat="task in tasks">
    <li>
      <span ng-class="{isDone: task.isDone}">{{task.title}}<
/span>
      <button ng-click="$parent.doneTask(task)" ng-show="!task.
isDone">
Done</button>
      <button ng-click="$parent.removeTask($index)">Remove</button>
    </li>
  </ul>
  <script src="js/app.js"></script>
  <script src="js/controllers/MainController.js"></script>
  <script src="js/services/TasksService.js"></script>
</body>
</html>
```

La première employée est `ng-app="todolist"` placée sur notre balise html. Celle-ci va nous permettre de spécifier que le module lié à la page est le module todolist, créé dans le fichier app.js cité plus haut. Ainsi, toute directive utilisée à l'intérieur de notre balise html sera contextualisée par notre module `todolist`.
Nous allons ensuite binder le contexte de la balise body grâce à la directive `ng-controller="mainController"`. Cette directive permet de contextualiser, tout comme `ng-app`, toutes les directives sous-jacentes et ainsi d'accéder aux propriétés et méthodes exposées via le paramètre `$scope` de notre contrôleur. Pour la suite, la directive `ng-model="taskTitle"` permet de réaliser un binding « two-way » sur la valeur de notre input. Ce type de directive permet de partager une donnée modifiable entre le modèle et la vue. Si l'utilisateur modifie la valeur de cet input, la propriété taskTitle de notre modèle sera automatiquement rafraîchie. La directive `ng-click="addTask()"` placée sur le bouton « + » permet de binder une méthode à exécuter lorsqu'un utilisateur cliquera sur celui-ci. Nous exécuterons donc la méthode addTask qui, créera une nouvelle tâche en invoquant la méthode `add` du service `tasksService`. Enfin, la directive `ng-repeat="task in tasks"` permet de dupliquer le contenu de la balise associée à celle-ci pour chaque élément contenu dans la propriété `tasks`. La syntaxe de la directive `ng-repeat` se rapproche de la syntaxe d'un foreach dans la plupart des langages. Nous allons donc parcourir la liste `tasks`, et pour chaque élément une variable locale `task` va être créée, et le corps de notre élément `<ul>` sera répété. Pour chaque `<li>` générée, la directive `ng-class ="{isDone: task.isDone}"` placée sur la balise `<span>`, permettant d'affecter des classes css à l'élément HTML en fonction de la valeur d'une expression, va ajouter la classe isDone si la tâche a été effectuée. Le contenu de cette balise est ensuite défini grâce à la syntaxe de templating d'AngularJS : `{{task.title}}`. La directive `ng-click="$parent.doneTask (task)"` du premier bouton va nous permettre d'exécuter la méthode doneTask du contrôleur en passant la tâche courante en paramètre. `$parent` va permettre de référer le contexte parent dans lequel nous nous trouvons, étant donné que nous sommes dans une balise ayant pour directive `ng-repeat`, il faut lui spécifier qu'il va falloir exécuter la méthode doneTask de notre parent (donc du contrôleur). Pour le bouton suivant, la démarche est similaire, à l'exception que le paramètre passé à la fonction n'est plus la tâche courante, mais l'index (`$index`) de la ligne.

Conclusion

Comme vous avez pu le découvrir, AngularJS vous sera d'une aide précieuse dans tous vos développements d'applications de type SPA en HTML / JavaScript. C'est un Framework qui vous fournira la structuration, autour de patterns, et les mécanismes, comme le binding, le templating ou la navigation, permettant de développer efficacement des SPA robustes. Site : http://angularjs.org/.

- Sébastien **OLLIVIER**, *Développeur*
- Pierre-Alexandre **GURY**, *Développeur*
- Julien **CORIOLAND**, *Tech Lead Web & Cloud*

INFINITE SQUARE

Exportation de fichiers SVG pour le web avec Adobe Illustrator CC

1ère partie

En tant qu'artiste et designer web, je suis ravi des possibilités que m'offre Illustrator CC en termes de gestion du format SVG (Scalable Vector Graphics) et de la souplesse qu'il me donne pour créer des graphiques de qualité. SVG étant un format vectoriel, je peux facilement créer des images qui s'adapteront automatiquement à tous les appareils modernes.

Les illustrations vectorielles sont très utiles pour les designers qui doivent créer des icônes, des symboles ou des logos en garantissant un rendu impeccable sur tous les écrans, y compris les écrans Retina. Outre une mise à l'échelle naturelle, les images SVG sont des fichiers de petite taille, dont la compression peut être optimisée par les serveurs web.

Si le format SVG suscite aujourd'hui un réel intérêt, il ne date pas d'hier : le consortium W3C a commencé à travailler dessus dès 1999 et recommande, depuis 2001, d'utiliser la version 1.1, la version 1.2 n'étant pas finalisée. Adobe a largement participé au développement de la spécification SVG. L'éditeur a publié un module externe avancé pour Illustrator, baptisé « SVG Viewer », qui prenait en charge la plupart des aspects de la spécification SVG. Il a toutefois cessé d'en assurer le support depuis quatre ans. Actuellement, SVG 1.1 est pris en charge par tous les navigateurs web sur les postes de travail et terminaux mobiles, à l'exception de Microsoft Internet Explorer 8 (et versions antérieures) et d'Android 2.3 (et versions antérieures). Le format SVG a le vent en poupe depuis l'explosion des terminaux mobiles et l'introduction de différentes densités d'écran.

Dans cet article, je vais vous présenter les options d'exportation à sélectionner pour obtenir des fichiers SVG de qualité pour vos sites web, tout en optimisant leur compression.

MULTIPLES OPTIONS D'EXPORTATION AU FORMAT SVG

Dans Illustrator CC (et CS6), il existe plusieurs façons d'exporter des images à partir des formes vectorielles que vous dessinez. Si vous sélectionnez Fichier > Exporter, vous avez le choix entre le format PNG, Flash, AutoCAD, TIFF et Photoshop, mais pas SVG. Intéressant... Si vous sélectionnez Fichier > Enregistrer pour le Web (sachant que vous ciblez des navigateurs web), vous pouvez choisir le format PNG, GIF ou JPEG. Aucune trace là non plus du format SVG.

Pour exporter des fichiers au format SVG depuis Illustrator, vous devez sélectionner Fichier > Enregistrer sous. Un fichier SVG est un fichier XML dans lequel Illustrator peut stocker des informations confidentielles à des fins de modifications ultérieures. Cela permet d'enregistrer un fichier Illustrator complet en XML. Notez l'option Utiliser les plans de travail dans la boîte de dialogue Enregistrer sous (voir Figure 1). Si vous avez dessiné 10 symboles dans votre fichier Illustrator, vous pouvez utiliser cette option pour créer un plan de travail par symbole et générer 10 fichiers SVG (un par symbole) en une seule opération.

Si vous ne maîtrisez pas les plans de travail dans Illustrator, consultez le didacticiel Scripting for Illustrator CS6 (Élaboration de scripts dans Illustrator CS6), dans lequel j'explique comment créer des plans qui s'adapteront automatiquement à votre contenu Fig.1.

Cliquez sur Enregistrer et examinez le panneau Options SVG (voir Fig.2).

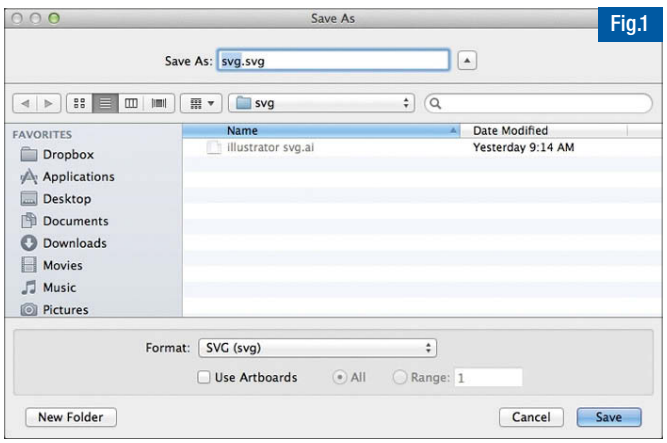


Fig.1

Enregistrement d'un fichier SVG depuis Illustrator CC

OPTIONS D'EXPORTATION RECOMMANDÉES POUR LE WEB

Les options sélectionnées dans la Figure 2 garantissent généralement d'excellents résultats :

- **Profil :** SVG 1.1
- **Polices :** Type — SVG ; Jeu partiel — Glyphes utilisés seulement (si vous utilisez une police spéciale pour votre illustration)
- **Emplacement de l'image :** Lier
- **Propriétés CSS :** Eléments de style
- **Positions décimales :** 1
- **Produire moins d'éléments <tspan> :** sélectionnez cette option
- **Utiliser l'élément <textPath> pour le texte curviligne :** sélectionnez cette option

Voilà ce qu'il faut définir pour exporter des fichiers SVG pour le web. Dans les sections ci-dessous, je vais détailler chacune de ces options.

Choix du profil SVG

Sélectionnez SVG 1.1, puisqu'il s'agit de la version recommandée par le consortium W3C (voir Fig.3). SVG Tiny avait été créé pour prendre en char-

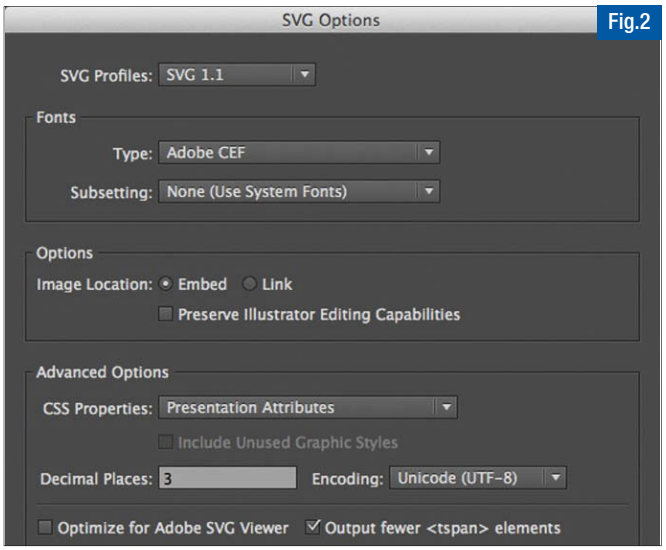


Fig.2

Options SVG recommandées pour exporter un fichier SVG pour le web

ge la première génération de terminaux mobiles, tels que les téléphones BlackBerry et les assistants personnels. Ce format est déconseillé si vous ciblez les smartphones.

Définition du type de police et du jeu partiel

Les designers web doivent sélectionner le type de profil SVG pour appliquer le format de police défini par le consortium W3C (voir Fig.4).

Le format SVG permet d'enregistrer un jeu partiel de glyphes dans le document. Certes, cela augmente la taille du fichier, mais c'est un moyen très pratique d'incorporer une police. Si vous ne prévoyez pas de modifier dynamiquement le texte de votre fichier SVG, vous pouvez sélectionner l'option Glyphes utilisés seulement. Illustrator détectera et incorporera automatiquement les caractères utilisés dans votre document.

Remarque : pour plus d'informations sur les diverses options de jeu partiel, consultez la rubrique Enregistrement d'illustrations de la documentation en ligne Illustrator.

La troisième option, Vectoriser, peut se révéler utile si vous ne voulez pas convertir automatiquement votre texte en tracés SVG. Vous pouvez la sélectionner pour une icône comportant peu de caractères, comme « Accueil » ou « Info ».

Que vous utilisiez un jeu partiel de glyphes ou convertissiez des polices en tracés SVG, la taille de votre fichier sera sensiblement la même.

En revanche, elle augmentera considérablement si vous utilisez des polices complexes (des polices manuscrites, par exemple). De même, si vous réduisez l'échelle d'une police convertie en tracés vectorisés, vous risquez de compromettre l'aspect visuel du contenu. (Tout cela reste purement théorique ; on ne constate pas de grande différence d'un navigateur à l'autre).

Remarque : d'après certains spécialistes, les polices SVG ne sont pas totalement prises en charge par Firefox et Internet Explorer, et les éditeurs de ces navigateurs n'envisagent pas d'y recourir. Donc, si vous ne voulez pas convertir vos polices en tracés vectorisés, utilisez des polices web. L'option Vectoriser augmente la taille de vos fichiers SVG et les rend moins accessibles. Les polices web constituent donc une solution plus intéressante. Si vous êtes en quête de polices web gratuites, consultez Adobe Edge Web Fonts, un sous-ensemble de polices Typekit gratuites.

Dans le fichier d'exemple, j'ai incorporé une police classique, Franklin Gothic, puis une police plus complexe, ChalkDust.

Sélection des options d'image

Vous pouvez choisir d'incorporer une image bitmap dans le fichier SVG, ou de créer un lien vers une ressource externe (voir Fig.5).

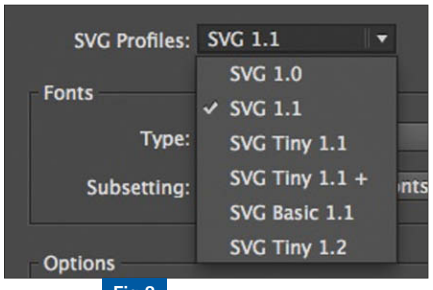


Fig.3 *Sélection du profil SVG*

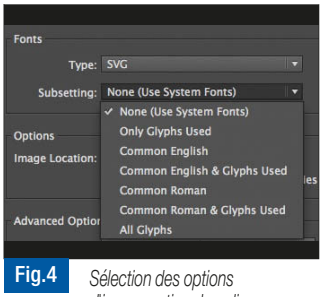


Fig.4 *Sélection des options d'incorporation de polices*

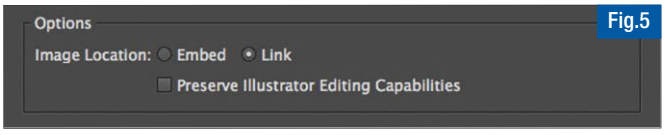


Fig.5 *Options d'exportation des images*

Voici le code SVG de l'image incorporée :

```
<?xml version="1.0" encoding="utf-8"?> <!-- Generator: Adobe
Illustrator 17.0.0, SVG Export Plug-In. SVG Version: 6.00
Build 0) --> <!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd"> <svg version
="1.1" id="homeElementChalk" xmlns="http://www.w3.org/2000
/svg"
xmlns:xlink="http://www.w3.org/1999/xlink" x="0px"y="0px"
width="105px" height="102px" viewBox="0 0 105 102" enable-back
ground="new 0 0 105 102" xml:space="preserve"> <circle fill
="#D86E27" cx="36" cy="36" r="36"/> <image overflow="visible"
width="480" height="468" xlink:href="
iVBORw0KGgoAAAANSUhEUgAAAeAAAAHUCAYAAADiP/ suAAAACXBIWXMAAA
sTAAALEwEAmpwYAAA4IGlUWHRYTUw6Y29tL mFkb2JlLnhtcAAAAAA
APD94cGFja2V0IGJlZ2luPSLvu78iIGlkPSJXNU0w TXBDZWhpSHpyZVN6T
lRjemtjOWQiPz4KPHg6eG1wbWV0YSB4bWxu czp4PSJhZG9iZTpucptZXR
hLyIgeDp4bXB0az0iQWRvYmUgWE1QIEN vcmUgNS41LWMwMTQgNzkuMTUxN
DgxLCAyMDEzLzAzLzEzLTEyOj A5OjE1ICAgICAgICAiPgogICA8cmRmOlJ
ERiB4bWxuczpyZGY9Imh0dH A6Ly93d3cudzMub3JnLzE5OTkvMDIvMjItc
mRmLXN5bnRheC1ucyMiP gogICAgICA8cmRmOkRlc2NyaXB0aW9uIHJkZjp
hYm91dD0iIgogICAgI CAgICAgICB4bWxuczp4bXA9Imh0dHA6Ly9ucy5hZ
G9iZS5jb20veGFw LzEuMC8iCiAgICAgICAgICAgIHhtbG5zOm1pPSJodHR
wOi8vcHVybC5v cmcvZGMvZWxlbWVudHMv (...) 9gBgdAY4OAztC/jCj1
BnDwBGB4Bjw9DOAC/8CHX2AGA4PpwahnYG fOFLqLNjAMPR0YlhaGeAF76E
OnsCMJwcHRiGtgZ84U+os3f7/wMAXxYhzahZKn4AAAAASUVORK5 CYII="
transform="matrix(0.1501 0 0 0.1501 32.94 31.7415)"> </image> </svg>
```

Et voici le code SVG d'un lien vers l'image :

```
<?xml version="1.0" encoding="utf-8"?> <!-- Generator: Adobe
Illustrator 17.0.0, SVG Export Plug-In. SVG Version: 6.00
Build 0) --> <!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd"> <svg version
="1.1" id="homeElementChalk" xmlns="http://www.w3.org/2000/
svg" xmlns:xlink="http://www.w3.org/1999/xlink" x="0px"
y="0px" width="105px" height="102px" viewBox="0 0 105 102"
enable-background="new 0 0 105 102" xml:space="preserve">
<circle fill="#D86E27" cx="36" cy="36" r="36"/> <image overflow
="visible" width="480" height="468" xlink:href="../../.. /
../../../../../Users/mchaize/Pictures/CC_Materials/CC_mnemo
nics/CC_mnemonics_PNG/ Ai_CC_mnemonic.png" transform="matrix
(0.1501 0 0 0.1501 32.94 31.7415)"> </image> </svg>
```

Cette option nécessite quelques corrections. Le lien vers une ressource externe est loin d'être la meilleure solution, mais le lien relatif pose aussi des problèmes car il pointe vers le fichier d'origine enregistré sur votre disque dur. Par ailleurs, si Illustrator génère un fichier JPEG, vous ne pouvez pas non plus contrôler la compression.

Le mieux est donc d'utiliser l'option Lier ; vous devez alors modifier le paramètre HREF afin qu'il pointe vers votre propre fichier d'image et compresser ce dernier via Adobe Photoshop CC.

La suite dans le prochain numéro.

Michaël **Chaize**

Basé à Paris, Michaël est un spécialiste de Creative Cloud chez Adobe. Il travaille essentiellement sur le web, les terminaux mobiles et les jeux.

Son blog : creativedroplets.com

Java Virtual Machine : optimisation et réactivité de la JVM 1ere partie

Parmi les grands axes d'optimisation d'une application Java, il y a la bonne gestion de sa mémoire et du garbage collector. Pour rappel, lorsque celui-ci s'exécute, toute l'application peut être figée pendant quelques millisecondes (ou plus), le temps de nettoyer les références inutiles. Le second axe d'amélioration est d'optimiser les accès au monde extérieur à la JVM (notamment dans le cadre d'accès à la base de données et des requêtes SQL / HSQL ou autres entrées/sorties).

Cet article est le premier d'une série d'articles autour d'une idée centrale : **la machine virtuelle Java optimise plus facilement un code court et concis.**

Pour optimiser une application Java, il ne faut pas négliger le code lui-même. Le bon vieux CheckStyle nous agace parfois mais il nous alerte sur la longueur maximale d'une méthode. A ce propos, le plugin Toxicity Chart pour Sonar, qui se base sur CheckStyle, peut être intéressant comme nous allons le voir. Parmi les bonnes pratiques de développement, il est primordial de ne pas faire des méthodes trop longues (30 lignes maximum) afin qu'elles puissent être visibles sur un seul écran.

Cette longueur aide non seulement à la lecture, à la compréhension et à la maintenance de cette méthode, mais également à accélérer l'exécution de celle-ci par une meilleure optimisation dynamique de la JVM. **Une méthode courte peut être jusqu'à 4400 fois plus rapide qu'une méthode trop longue non compilable.**

C'est là que la qualité de code rejoint la vitesse d'exécution.

Résumé : comment gagner en rapidité en optimisant la JVM

L'optimisation dynamique concerne tous les langages s'exécutant dans la JVM : Java, Scala, Groovy, Jruby, Closure, Javascript, mais aussi tout ce qui passe par une création de classe Java intermédiaire : JSP, XSD (via xjc), WDSL (via wsimport).

Pour que les méthodes de la JVM soient transformées en code assembleur machine par la JVM, il faut 2 conditions essentielles :

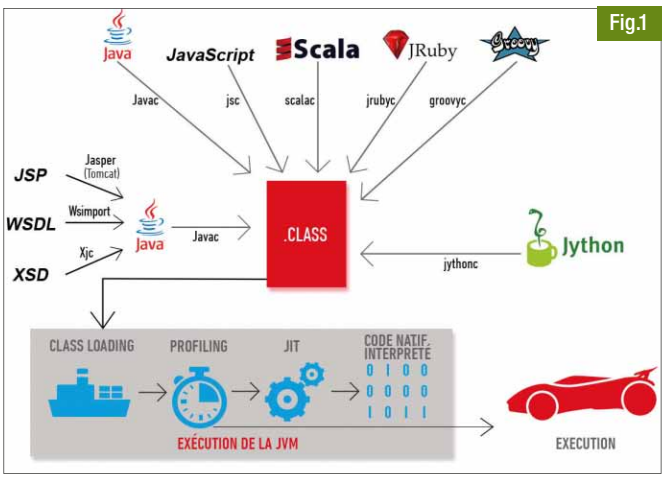
- Un **nombre d'exécutions** de cette méthode qui doit être au moins de 1500 fois pour le mode -client ou de 10000 fois pour le mode -server.
- Une **taille de méthode** inférieure à 8Ko de bytecode (difficile de donner un équivalent en nombre de ligne de code). Une taille supérieure à 8Ko empêche toute compilation en assembleur de cette méthode.

A noter que l'optimisation du code par le compilateur JIT (Just In Time) et la traduction du bytecode en assembleur est d'autant plus grande si l'inlining est activé. L'inlining est la fusion d'une méthode appelante et appelée afin d'avoir une vision globale du traitement et d'en faciliter l'optimisation. Une méthode dépassant les 1000 octets de bytecode ne pourra pas profiter d'inlining d'inclusion. Elle sera donc moins rapide même si une partie de son code sera transformée en assembleur. La JVM utilise différents type d'inlining, nous y reviendrons plus tard.

Par défaut, les JVM 32bits activent le mode -client alors que les JVM 64bits ne proposent que le mode -server. Selon les traitements, il y a un rapport de 1 à quasiment 800 entre le mode -client et -server pour la même JVM.

A noter que dans des calculs intensifs, un rapport de 1 à 4 peut aussi exister entre un JDK 32bits et 64bits.

Fig.1



QUAND CLARTÉ DE CODE RIME AVEC RAPIDITÉ D'EXÉCUTION

Jusqu'à quel point le code assembleur généré est optimisé par rapport à un compilateur C

La transformation du bytecode en code assembleur machine apporte un réel gain de vitesse. Pour s'en convaincre, et pour l'exemple, le calcul d'une valeur de la suite de Fibonacci compilée en C avec l'optimisation agressive -O3 a un même temps de réponse moyen que la JVM 1.7 d'IBM utilisant un cache de méthodes déjà compilées (via –Xshareclasses). L'écart est seulement de 31 % avec le JDK 8 64bit (sans option particulière au lancement de la JVM).

version C

```
#include <stdio.h>
#include <time.h>

unsigned int fib(unsigned int n)
{
  unsigned int fibminus2=0;
  unsigned int fibminus1=1;
  unsigned int fib=0;
  unsigned int i;
  if (n==0 || n==1) return n;

  for(i=2;i<=n;i++){
    fib=fibminus1+fibminus2;
    fibminus2=fibminus1;
    fibminus1=fib;
  }

  return fib;
}
```

version Java

```
class fib_nc {

  public static long fib(long n)
  {
    long fibminus2=0;
    long fibminus1=1;
    long fib=0;
    long i;
    if (n==0 || n==1) return n;

    for(i=2;i<=n;i++){
      fib=fibminus1+fibminus2;
      fibminus2=fibminus1;
      fibminus1=fib;
    }

    return fib;
  }
```

```c
int main(int argc, char **argv)
{
  unsigned int n;
  if (argc < 2) {
    printf("usage: fib n\n"
      "Compute nth Fibonacci
number\n");
      return 1;
  }
  clock_t t1, t2;
  t1 = clock();

  n = atoi(argv[1]);
  printf("fib(%d) = %d\n", n,
fib(n));
  t2 = clock();
  int diff = (((float)t2 -
(float)t1));

  printf("time : %d ms",diff)

  return 0;
}
```

```java
  public static void main(String[] args)
  {
    long n;
    if (args.length < 1) {
      System.out.println("usage:
fib n\nCompute nth Fibonacci
number\n");
      return;
    }
    long t1, t2;
    t1 = System.currentTimeMillis();

    n = Long.valueOf(args[0]);
    System.out.printf("fib(%d)
iteratif = %d\n", n, fib(n));
    t2 = System.currentTimeMillis();

    System.out.println("time
in ms : " + (t2-t1));

  }
}
```

NB : La ligne de commande à exécuter pour compiler sous Mingw32 est la suivante sous Windows : gcc fib.c -o fib.exe -O3.

Du code source à l'optimisation dynamique

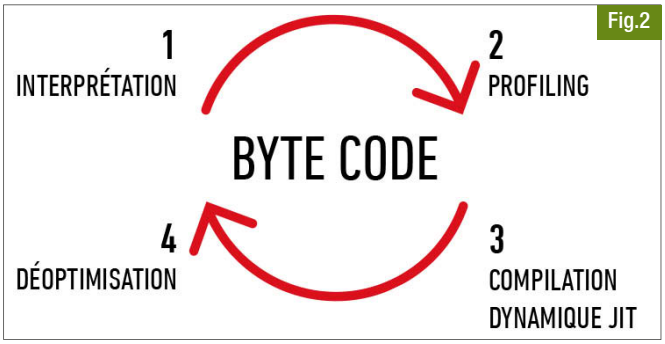
La JVM est une machine virtuelle multi-langage et le nombre de langages supportés n'a pas cessé de grandir. Certains langages se limitent à de l'interprétation (même si elle est plus rapide sous Java 7 grâce au InvokeDynamic) mais les plus utilisés proposent également un compilateur pour passer du monde Scala, Groovy, Jruby, Jython, Javascript à un fichier .class en bytecode.

De plus, d'autres outils proposent de générer des classes Java qui à leur tour s'exécuteront dans la JVM. Les JSP sont transformées en servlet .java via jspc (ou jasper sur tomcat) puis compilées en .class.

Pour les Web Service Soap, avec les fichiers XSD et WSDL, ce sont xjc et wsimport qui génèrent des classes Java et .class à partir du XML. Il y a aussi des bascules entre JSON et Java.

Enfin, il y a tout ce qu'on pourrait générer via de la génération de code maison avec du velocity et du freemarker. Le pire, c'est que ce code généré est très souvent ignoré dans les analyses de code statique de type Sonar alors que la qualité du code généré ne lui permet pas toujours d'être optimisée par le compilateur JIT Fig.1.

A la compilation en bytecode, aucune optimisation particulière n'est effectuée. C'est au moment de l'exécution qu'une optimisation dynamique sera réalisée sur les classes et les méthodes en ayant le plus besoin. **Au début du lancement de la JVM, toutes les méthodes sont interprétées** jusqu'à ce qu'un seuil en nombre d'exécutions soit atteint, et, à condition que le code de la méthode à compiler ne soit pas trop long. **C'est là qu'intervient le compilateur JIT (Just In Time) qui compile en assembleur pour le processeur hôte (x86 ou x64 pour Intel) les instructions de bytecode** Fig.2.



Sur Android, le passage du mode interprété à un compilateur JIT a vu aussi une amélioration de performance, qui selon les benchmarks, passait sur le même appareil d'un facteur de 6 à 600 selon les tests dont en voici un sur le Nexus One (2010) lors du passage d'Android 2.1 à 2.2.

Sur les navigateurs internet, ce compilateur JIT est également en place depuis quelques années pour que le code javascript soit lui aussi traduit en code assembleur. Firefox utilise le moteur IonMonkey qui possède un compilateur JIT. Quant à Chrome avec le moteur V8, il utilise un autre moyen, la compilation dite "**Ahead Of Time**" (AoT) où toutes les classes Javascript sont compilées en assembleur à leur première exécution.

Ce type de compilation AoT est également possible avec la JVM Oracle avec l'option **-Xcomp**. Cependant, nous y reviendrons plus tard, cela ajoute un temps de démarrage plus long et des optimisations précoces.

D'autres options sont possibles au lancement de la JVM :

- Xint : pour un mode totalement interprété sans aucune traduction en assembleur par le JIT ;
- Xmixed : qui est un compromis entre le tout compilé et le tout interprété. C'est aussi le mode par défaut de lancement de la JVM.

A noter que sur Android 4.4 (Kitkat), un nouveau mode d'exécution Java est disponible (mais non activé par défaut) : ART en remplacement de la JVM Dalvik. Ce que fait ART est une compilation Ahead Of Time au moment de l'installation de l'application.

Les types d'optimisations disponibles de la JVM

Afin d'optimiser au mieux le code interprété (puis compilé en assembleur), la JVM a besoin d'obtenir des métriques sur les méthodes et ainsi de faire un profiling sur l'ensemble des méthodes des classes chargées.

Voici un aperçu avec 9 types d'optimisations différentes parmi plus de 64 proposées par la JVM de Java 7 :

- **Inlining :** c'est la fusion du code d'une méthode appelante avec le code de la méthode appelée. Il en existe 4 types différents, mais le plus courant est le "monomorphic inlining" : ex: System.currentTimeMillis() lors d'appel de méthodes statiques
- **Escape analysis :** analyse le code en profondeur et, lorsque c'est possible, évite de créer des instances d'objets lorsque celles-ci pointent sur des valeurs qui ne bougent pas et remplace l'expression par l'utilisation d'une constante.
- CHA **Class Hierarchy Analysis** (Analyse des héritages et du polymorphisme qui permet une dévirtualisation) **:** permet si besoin de dévirtualiser une méthode polymorphe héritée. Cela permettra d'utiliser de l'inlining sur cette méthode dévirtualisée.
- **Optimisation CPU** x64, MMX, SSE **:** ajout dans le code assembleur des instructions x64 sur processeur 64bits, et des optimisations MMX et SSE pour les calculs sur les types « float » et « double » et également la recopie de tableaux. *Après divers essais, SSE et MMX accélère surtout les JVM 32bits car cela leur met à disposition des registres 64bits, en passant à des JVM 64bits, l'accélération est moins visible.*
- **Loop unrolling :** lorsqu'une boucle s'effectue sur peu d'éléments et de manière constante, l'optimiseur remplace la boucle par n-fois le corps de cette boucle.
- **Lock coarsening :** lors d'une boucle où une méthode synchronisée est appelée n-fois (avec le mécanisme de verrou aussi appelé n-fois), l'optimiseur replace les « n » synchronized par un seul avant la boucle principale, ce qui optimise les ressources de verrous.
- **Lock eliding :** élimine un bloc "synchronized" lorsque celui-ci n'est pas nécessaire

- **Dead code elimination :** élimination du code mort (ex: condition irréalisable)
- **Duplication code elimination :** suite à une première optimisation, simplification du code optimisé

La JVM a deux types de stratégies correspondant à 2 compilateurs JIT différents :

- Démarrage rapide et optimisation moyenne : c'est le -client ou Compilateur C1
- Profiling plus long des méthodes mais meilleure optimisation : c'est le mode -server ou Compilateur C2

Sur le JDK Oracle, un nouveau mode a fait son apparition depuis Java 6, c'est le mode Tiered (nivelé en français). Il tente de regrouper les qualités des 2 compilateurs C1 (-client) et C2 (-server). Après des benchmarks, les résultats que j'ai obtenus ne sont pas assez probants. On peut jouer sur les paramètres suivants : -XX:+TieredCompilation ; seul ou en ajoutant -XX:TieredStopAtLevel=1 (jusqu'à 4 qui est la valeur par défaut).
Autre type de compilation : l'**OSR** (On Stack Remplacement). C'est une compilation à la volée lors d'une boucle sur une méthode encore en bytecode avec un nombre élevé d'itérations. C'est le coeur de la méthode qui est alors transformé en code assembleur juste pour l'exécution en cours. C'est une compilation "éphémère" mais qui peut accélérer des boucles répétées un nombre de fois très élevé.
Sinon, alors que côté Microsoft et sa CLR, il a fallu attendre le framework .net 4.5 pour avoir la compilation assembleur en tâche de fond, celle-ci est disponible par défaut depuis Java 5.
Un dernier moyen d'optimiser le temps de lancement de la JVM et de compilation est d'avoir déjà sous la main le code assembleur des classes du noyau (RT) de la JVM, c'est ce que fait le JDK Sun puis Oracle depuis la version 5 via l'utilisation des classes JSA (-Xshare). Par défaut, le système tente de les utiliser et continue à fonctionner tant qu'il ne parvient pas à le faire (l'activation se fait par -Xshare:dump ou –Xshare:on).

La JVM IBM j9 utilise aussi un moyen équivalent via l'option de démarrage -Xshareclasses.
Avec Java 8, est prévu également le multi-tenant dont le but est de centraliser les classes communes entre applications tournant dans la JVM et aussi de n'avoir qu'une seule copie des méthodes transformées en assembleur par le JIT et ainsi économiser du temps de compilation et de la mémoire. Cependant l'utilisation de debug, de profiling et d'agent (instrumentation) est problématique dans ce mode.

A noter que dans le monde des machines virtuelles, Microsoft avec la CLR a aussi de la compilation "AoT" via l'outil ngen qui propose une version du bytecode (IL) compilée en assembleur mais sans profiling et donc avec une optimisation moindre.

Une autre solution proposée par Microsoft qui n'existe pas encore dans la JVM est MPOG : Managed Profiled Guided Optimization dont le but est de réconcilier les 2 mondes JIT et AoT via la création d'un profiling JIT pendant l'exécution de l'application puis une compilation "Ahead Of Time" prenant en compte les informations de profiling lors de la phase précédente. Le gain de performance pourrait être de 25% par rapport à du "simple" AoT via ngen. La suite dans le n°178.

Olivier **Guilloux** - *SQLi*

Déployez rapidement un site de partage de média en NodeJS avec AWS Elastic Beanstalk

Dans cet article nous allons détailler les étapes de la création d'un site de partage de média (des vidéos, images, documents PDF...) qui respecte les attentes des utilisateurs. Ainsi, le site devra être capable de fonctionner en toutes circonstances, tout en supportant un nombre grandissant d'utilisateurs et en offrant un moyen d'authentification simple. Les fonctionnalités qui seront mises en place seront les suivantes : authentification des utilisateurs, upload des fichiers média, diffusion et partage de ces fichiers de manière simple et rapide.

Le succès d'un tel site est difficilement prédictible, il est donc primordial de s'assurer, dès son lancement, qu'il pourra encaisser un pic de fréquentation. C'est pourquoi le langage de développement du site sera Node.js, connu pour ses capacités de scalabilité, mais également pour son côté asynchrone et non bloquant, tout à fait adapté aux applications web à forte fréquentation.

Nous utiliserons les services Amazon Web Services pour déployer ce site : le service IAM (AWS Identity and Access Management) afin de gérer le fédération d'identité web pour l'authentification, S3 pour stocker les fichiers média et enfin Elastic Beanstalk pour déployer notre application Node.js.

Toutes les informations figurant dans le présent article (y compris les éléments de code) sont mises à disposition à des fins strictement pédagogiques. Nous vous recommandons fortement de ne pas utiliser les éléments de code présentés dans l'article sans modification, comme la personnalisation et l'ajout d'éléments de sécurité.

Il convient, en effet, de rappeler qu'Amazon Web Services propose un modèle de responsabilité partagée où AWS assure la sécurisation des couches basses (bâtiments physiques, réseaux, virtualisation...) – et où le client est responsable de la sécurité de son application, comme dans l'exemple que nous allons passer en revue aujourd'hui.

Etape 1 : Premier déploiement sur Elastic Beanstalk

Elastic Beanstalk est un service managé de serveur web intégrant nativement la redondance et la scalabilité. Il supporte les langages suivants : Java, .NET, PHP, Python, Ruby, l'utilisation de Docker et bien évidement Node.js.

Il permet en quelques clics de déployer une nouvelle application ou une nouvelle version de celle-ci soit au moyen de la console en ligne, soit avec des lignes de commande, ou encore via des plug-in pour Eclipse, Visual Studio et GIT.

Grâce à Elastic Beanstalk, vos applications sont en ligne en quelques minutes, sans que vous ayez besoin de vous occuper des opérations d'infrastructure, cela vous laissant donc du temps pour des activités différenciatrices sur votre segment de marché.

Créez le squelette de votre application dans un nouveau répertoire grâce au framework Express.

Modifiez le fichier app.js pour créer le serveur Node.js, Elastic Beanstalk se chargera ensuite de le lancer au démarrage.

```
> express -H mon-application && cd mon-application && npm install
> echo «app.listen(process.env.PORT || 3000);» >> app.js
```

Initialisez ensuite le répertoire Git et faites votre premier commit du squelette :

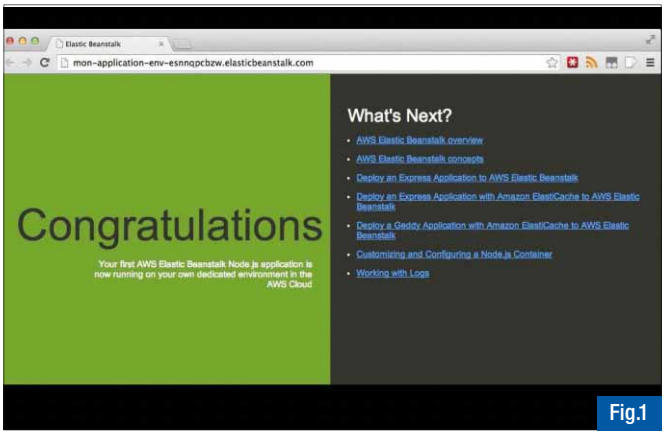


Fig.1

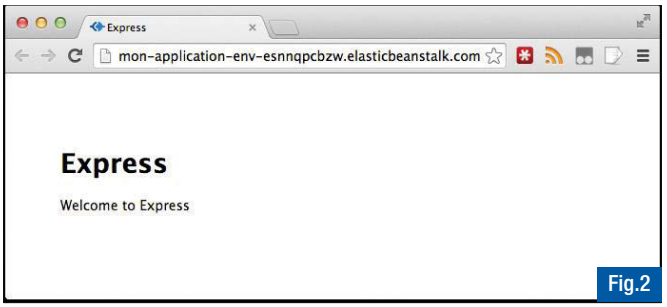


Fig.2

```
> git init && git add . && git commit -am "initial commit"
```

Initialisez ensuite l'environnement Elastic Beanstalk à l'aide de l'outil en ligne de commande « eb » et sélectionnez un déploiement de type web-server, avec un load balancer, sans base de données, sur la région EU-West 1 (Irlande) (le choix de la région vous appartient, le processus sera le même sur les 9 régions accessibles dans le monde)

```
> eb init
```

Votre environnement est prêt à être lancé pour accueillir votre application. Tapez la commande suivante pour lancer votre environnement dans le Cloud AWS sans déployer votre premier commit :

```
> eb start
```

Cette opération peut prendre quelques minutes car toutes les ressources de l'environnement sont créées pour la première fois. Une fois l'environnement déployé, eb vous fournira l'URL de votre environnement. En tapant cette URL dans votre navigateur, vous devriez voir ceci : Fig.1.

Il ne reste plus qu'à déployer votre application Express à l'aide de git avec la commande suivante :

```
> git aws.push
```

Une fois l'application déployée, si vous rafraichissez la fenêtre de votre navigateur, vous devriez voir ceci : Fig.2.

Etape 2 : Fédération d'identité web avec IAM

Avec IAM, vous pouvez autoriser vos applications basées sur un navigateur à accéder de façon sécurisée aux ressources AWS en demandant des identifiants de sécurité temporaires qui n'accordent l'accès qu'à des ressources AWS spécifiques, comme par exemple, ici, un espace de stockage sur Amazon S3.

Pour mettre en œuvre la fédération d'identité vous pouvez utiliser l'utilitaire ligne de commande unifiée (http://aws.amazon.com/cli/) pour créer un profil d'utilisateur. C'est ce profil qui portera les droits utilisateur à chaque connexion. La fédération d'identité proposée par IAM reconnaît trois fournisseurs d'identité sur internet :

- Amazon.com
- Google
- Facebook

Pour notre exemple nous avons choisi de confier l'authentification des utilisateurs à Amazon. En premier lieu rendez-vous sur http://login.amazon.com pour créer votre application en quelques clics, vous pourrez ensuite accéder aux informations suivantes : Fig.3.

Il suffit ensuite de créer un profil utilisateur dont la stratégie d'authentification est déléguée à Amazon.com à l'aide de la commande suivante :

```
aws iam create-role –role-name mon-application-role –assume
-role-policy file://mon-application-role-trust-policy.json
```

Le fichier `mon-application-role-trust-policy.json` est un simple fichier texte au format JSON décrivant la méthode de délégation d'authentification. Il se présente comme ceci :

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Principal": {
        "Federated": "www.amazon.com"
      },
      "Effect": "Allow",
      "Condition": {
        "StringEquals": {
          "www.amazon.com:app_id": "amzn1.application.
123exemple456"
        }
      },
      "Sid": ""
    }
  ]
}
```

Si nous avions confié l'authentification des utilisateurs à Google ou Facebook, il aurait fallu remplacer «www.amazon.com:app_id»:»amzn1.application.123exemple456» dans ce code par l'identifiant du fournisseur d'identité choisi.

Au niveau applicatif nous avons utilisé le module « passport » pour intégrer la fédération d'identité à notre serveur Node.js, ce qui nous amène au résultat suivant : Fig.4 et 5.

Etape 3 : Upload et partage de médias

Nous avons donc, en quelques lignes de code, monté de toutes pièces les bases de notre application de partage de média. Nous avons commencé par l'authentification, ce qui nous permet maintenant d'aborder l'upload sécurisé des médias à partager.

Pour ce faire nous allons utiliser Amazon S3, service de stockage pour Internet. S3 offre une interface publique de type REST qui permet de stocker et d'extraire des données, de manière sécurisée, à tout moment et depuis n'importe quel accès Internet, quel qu'en soit le volume : ce service semble donc tout indiqué pour héberger un volume de média large et grandissant.

De plus, nous pouvons profiter du fait que S3 propose une interface publique : ainsi au lieu d'uploader les fichiers média vers notre application, nous pouvons directement les envoyer vers S3. Cette stratégie est intéressante à plusieurs titres :

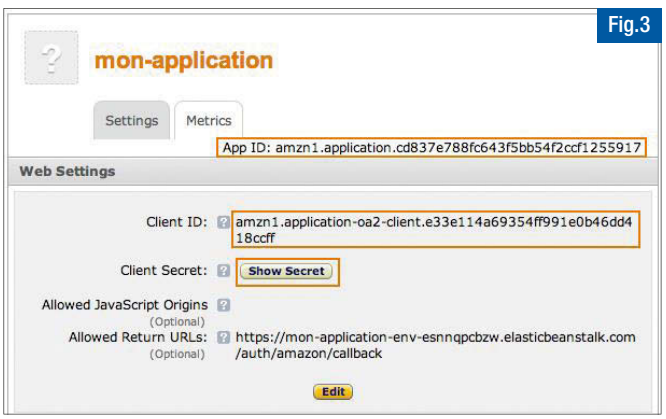


Fig.3

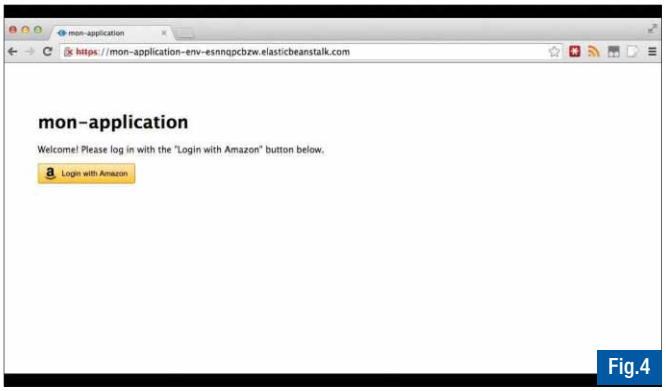


Fig.4

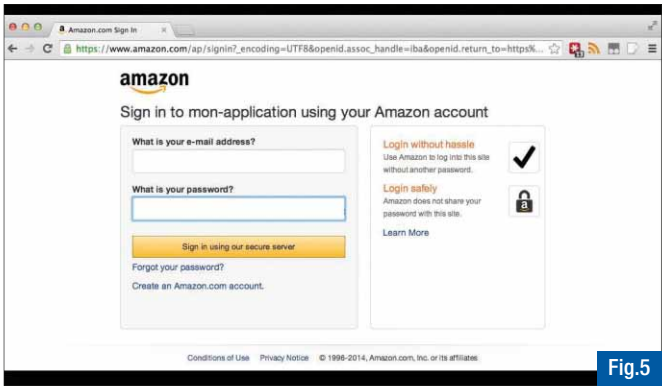


Fig.5

- Vous n'avez pas de code supplémentaire à déployer sur votre application pour la réception et le stockage des fichiers uploadés,
- Les opérations de réception et stockage se faisant sur Amazon S3, votre consommation CPU et bande passante sont réduites d'autant, limitant ainsi vos coûts d'infrastructure,
- La disponibilité et la scalabilité sont directement prises en charge par le service Amazon S3 : vous n'avez pas besoin de vous en charger.

Pour mettre en œuvre l'opération d'upload, il est nécessaire de :

- Créer un espace de stockage Amazon S3 dédié à l'application,
- Modifier la page d'index pour y faire apparaître, lorsque l'utilisateur est authentifié, un formulaire d'envoi assorti d'un champ `file`,
- Obtenir de l'application back-end le droit d'envoyer un fichier sur S3, sur la base de l'identité de l'utilisateur connecté et authentifié,
- Ajouter un chemin à l'application web pour qu'elle reçoive la notification de fin de l'upload.

Commençons donc par créer un `bucket` Amazon S3. Les buckets sont les espaces de stockages unitaires de Amazon S3. Vous pouvez créer le bucket avec la commande suivante :

```
> aws s3 mb s3://mon-application-bucket –region eu-west-1
```

Les Buckets Amazon S3 sont créés par défaut avec un niveau de protection élevée : aucun autre utilisateur, à part le créateur du bucket, n'a de droit sur les fichiers qui s'y trouvent. Grâce à la fédération d'identité, nous allons pouvoir autoriser les utilisateurs authentifiés de notre application à déposer des fichiers sur le nouveau bucket. Ajoutez un fichier de règles de sécurité sur le rôle IAM grâce à la commande suivante :

```
> aws iam put-role-policy –role-name mon-application-role –
policy-name mon-application-role-policy-s3 –policy-document
 file://mon-application-role-policy-s3.json
```

Où le fichier `mon-application-role-policy-s3.json` contient la description de la stratégie de sécurité. En particulier les lignes suivantes (extrait):

```
"Resource": [
  "arn:aws:s3:::mon-application-bucket/users/${www.amazon.com:
user_id}",
  "arn:aws:s3:::mon-application-bucket/users/${www.amazon.com:
user_id}/*" ]
```

Ajoutons maintenant un formulaire d'upload sur la partie serveur Node.js, cela permettra aux utilisateurs d'uploader des fichiers média directement sur Amazon S3.

```
 <form action="https://mon-application-bucket.s3.amazonaws.
com" method="post" enctype="multipart/form-data" id="myform"
 role="form">
    <input type="hidden" name="AWSAccessKeyId" id="fld_AWSAccessKeyId">
    <input type="hidden" name="acl" id="fld_Acl">
    <input type="hidden" name="key" id="fld_Key">
    <input type="hidden" name="policy" id="fld_Policy">
    <input type="hidden" name="redirect" id="fld_Redirect">
    <input type="hidden" name="signature" id="fld_Signature">
    <input type="hidden" name="x-amz-security-token" id="fld_Token">
    <label for="file">Please select a file to upload</label>
    <input type="file" id="file" name="file">
    <input type="submit" name="monbouton" id="btn_submit"  class
="btn btn-default">
    </form>
```

Fig.6.

Une fois l'upload terminé l'application affiche un lien pour venir télécharger le fichier média depuis Internet Fig.7.

Etape 4 : A vous de jouer !

Pour ceux d'entre vous qui souhaitent déployer ce site eux-mêmes, un tutorial complet est disponible à l'adresse suivante : http://bit.ly/awsmedia. Il vous suffira de créer votre propre compte AWS pour déployer les ressources nécessaires, et de suivre les instructions données. Vous aurez ainsi accès à l'intégralité du code source.

Et après ?

Vous êtes maintenant en mesure d'héberger un site de partage de média sur base d'une architecture scalable et hautement disponible.

Vous pouvez donc, dès maintenant, enrichir ce site web de partage, en ajoutant des fonctionnalités supplémentaires. Par exemple pour les fichiers médias de type vidéo, vous pouvez intégrer un lecteur directement sur le site, ou encore transcoder automatiquement une vidéo uploadée par un utilisateur dans plusieurs formats afin de la lire sur une large majorité de terminaux mobiles. Ces points pourront faire l'objet d'un autre article 'pas à pas'.

En attendant, n'hésitez pas à modifier le code existant pour le personnaliser ou pour déployer de nouvelles applications sur Elastic Beanstalk.

Michael **Garcia**
Solutions Architect, AWS

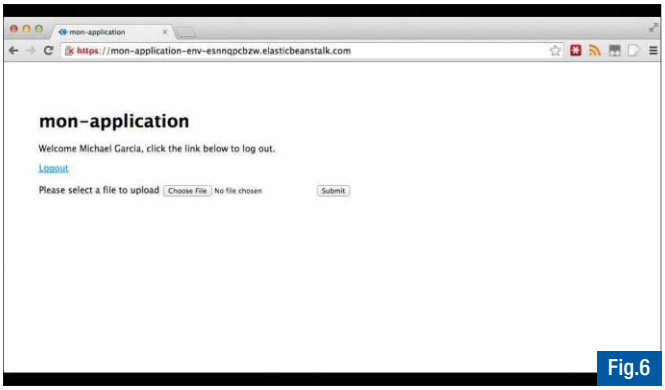


Fig.6

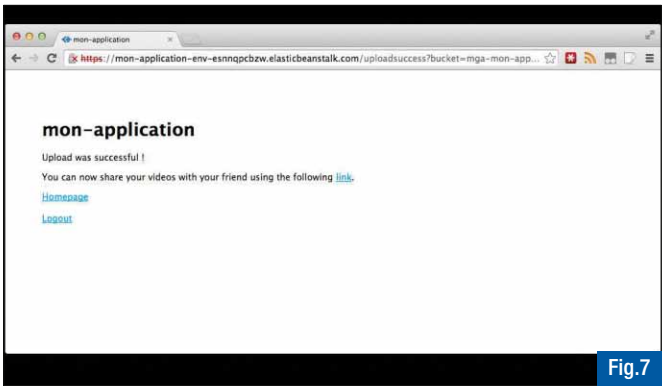


Fig.7

3D temps réel sur mobile ou le recours aux meilleures astuces

La 3D temps réel, c'est très bien. Si vous lisez ces lignes portant sur BabylonJS, je ne vous apprends rien. On commence à maîtriser des aspects techniques très pointus qui sont dignes des derniers moteurs professionnels (UDK, IDtech5, CryEngine 3) des productions cinématographiques. A mesure que ces technologies haut de gamme progressent, les limites basses sont aussi repoussées, avec de vraies solutions sur navigateurs, accompagnées de solides technologies, comme le WebGL et BabylonJS.

Mais tout comme aux premiers temps de la 3D temps réel, les designers et développeurs essayaient de trouver des palliatifs et des raccourcis techniques pour imiter les fonctionnalités avancées des images de synthèse, avec beaucoup moins de ressources. La mainmise des appareils mobiles sur le marché du numérique grand public pousse aujourd'hui à nous reposer la question de la performance sur un environnement limité. Et vous ne trouverez pas ici une liste exhaustive des trucs et astuces que je pourrais employer, mais plutôt la réponse que j'ai apportée au besoin d'optimisation d'une scène 3D donnée, pour la porter d'un moteur « full power- DirectX10 – gros PC » vers le monde de la mobilité et du web.

Un état de l'Art mobile.

Aujourd'hui, le jeu sur téléphone, secteur pour le moins dans la cible d'un moteur tel que BabylonJS, c'est souvent synonyme de jeu en 2D, ou en 3D propriétaire, ultra-optimisée. Et pour cause: un appareil mobile haut de gamme actuel tel un Nokia 1520 embarque 2 GO de mémoire vidéo de type DDR3, alors qu'un PC haut de gamme oscille entre 6 et 12 Go de DDR5 (un rapport de 3 à 6, donc). Un téléphone, c'est peu ou prou 1 Go de Ram spécifique basse consommation, un PC, c'est entre 8 et 16 Go (16 fois plus). Et pour en finir avec les comparaisons un peu brutales, une carte vidéo haut de gamme, c'est 5700 cœurs logiques, tandis que sur un téléphone, nous n'en trouvons que 72. Vous l'aurez compris, on ne fera pas tenir facilement un ensemble d'assets prévus pour une représentation 3D PC sur un téléphone, fut-il gonflé aux hormones. Sans compter qu'à ces contraintes de performances s'ajoutent les exigences de la mobilité. Charger une scène de 40 Go est simplement inenvisageable en 4G. Il faut donc intégrer des systèmes de chargements sélectifs, et optimiser ses données et ressources pour qu'elles puissent transiter rapidement sur les ondes.

Du sang, de la sueur et des larmes

Pour illustrer la somme d'efforts d'optimisation requise pour convertir une scène « high-end » en une scène jouable sur téléphone ou sur tablette, j'ai demandé à des partenaires de longue date, la société Enozone (www.enozone.net), spécialisée dans les maquettes virtuelles, la réalité augmentée et les maquettes virtuelles, de bien vouloir me confier leur scène 3D la plus emblématique afin de me permettre de la convertir vers BabylonJS. J'ai donc reçu la scène « Hill Valley », reprenant la charmante place du centre-ville américain bien connue des afficionados de la trilogie « Retour vers le futur ». Clou du spectacle, la GMC DeLorean de la série se trouve dans la scène.

On se remonte les manches

La vraie question à se poser, lorsque l'on charge la scène sur un PC bien vitaminé est la suivante : comment je vais bousiller élégamment le boulot d'Enozone, afin de faire rentrer la scène dans un téléphone. Car, en effet, il va falloir faire des choix drastiques, amputer les ressources et dégrader les assets, en conservant un visuel flatteur, en fonction de l'usage et de la taille de l'écran ciblé. Pour cela j'utilise un logiciel connu du marché, le célèbre 3d studio Max d'Autodesk, mais les techniques décrites plus loin sont, en général, disponibles dans tous les outils destinés aux artistes 3D.

De la même manière, en ce qui concerne les textures, je me sers de Photoshop d'Adobe, qu'on ne présente plus. C'est d'ailleurs avec lui que nous commençons cette phase d'optimisation. En effet, la première étape consiste à réduire la taille des textures utilisées dans la scène. Pour mémoire, les textures sont employées pour figurer les aspects des objets de la scène 3D (diffus, ambiant, spéculaire, réflexion, relief, etc...). On utilisera comme châssis une taille de

texture multiple de 8, pour convenir à la mémoire des appareils. Ainsi, hors compression, un fichier image de 256x256 pixels fait environ 200 ko alors qu'un fichier de 128x128 px fait 50 ko, soit 4 fois moins. Attention toutefois à ne pas réduire trop drastiquement pour ne pas trop dégrader le rendu visuel. Il faudra se poser constamment la question de la pertinence de la texture sur un écran de 4, 5 ou 6 pouces (comment est perçue telle ou telle texture sur un petit écran lorsque mes doigts sont dessus ?).

On veillera à utiliser un seul et unique format de texture (png ou dds), et on réduira la pollution et le « bruit » des textures afin que la compression soit la plus efficace possible (moins il y a de nuances et de dégradés, plus fort on peut compresser). On veillera également à intégrer les canaux d'opacité des shaders (les effets de transparence des matériaux des objets), directement dans le canal alpha des textures. Cela m'a permis de passer d'un total de 450 Mo de fichiers de textures à moins de 70 Mo. Avec un gain de performances certain, mais surtout un gros gain de temps sur celui du téléchargement.

Gardons aussi à l'esprit, en aparté, qu'aujourd'hui de nombreux moteurs tirent parti de textures procédurales, qui ne pèsent pratiquement rien et ne sont pas dépendantes de la résolution des assets, puisque dépendantes de formules mathématiques (voir http://www.allegorithmic.com/ pour plus d'informations à ce sujet).

Les matériaux.

On passe ensuite au niveau supérieur en allégeant systématiquement les matériaux, processus entamé alors que nous réunissions les textures de diffus (l'aspect d'un objet) et son éventuelle transparence (opacité) au sein d'une même texture. De la même manière, on essayera de faire correspondre sur un seul canal le maximum d'informations d'aspect.

Par exemple, l'artiste 3D utilise souvent un canal de diffus et un canal d'ambiant (ombrage) séparé pour plus de souplesse, et généralement sur deux coordonnées de textures différentes (l'un en mode planaire, l'autre en « aplati » ou « baked »).

Le mieux est ici de trouver le juste compromis permettant de faire correspondre ces deux textures à une seule et unique ressource. Cela permet de gagner une ou plusieurs textures et autant de coordonnées additionnelles.

Dans la même veine, et pour des soucis de confort, un artiste peut avoir recours à un « multi matériau ». Celui-ci permet d'affecter un matériau à un ensemble de faces d'un objet 3D, tandis qu'un autre ensemble recevra un autre matériau. Utile quand il s'agit de traiter un objet complexe. Un bâtiment par exemple peut être composé de faces portant un matériau de verre, un autre pour les briques, etc...

Chaque groupe de face contenant son matériau, mais aussi ses coordonnées de textures distinctes. Dans l'optique de faciliter la vie à notre moteur de rendu, on se focalisera sur la réduction du nombre d'entités à traiter. Le but est ici d'obtenir le sacro-saint « un matériau par objet ». La technique utilisée est celle du rendu sur texture, car elle va permettre à l'artiste de créer une image composite du shader d'origine contenant toutes les textures le composant. Ceci va permettre de réduire le nombre d'assets utilisés, mais aussi la taille du fichier descriptif (moins de coordonnées, moins de choses à assigner, moins de complexité pour le CPU).

Géo mais tri ?

Bien entendu, et même si c'est moins important que par le passé, on fera aussi une passe d'optimisation sur les géométries de la scène. Dans Hill Valley, ce travail a porté sur les végétaux, pour lesquels j'ai supprimé 25 % de faces sans trop dégrader le visuel. Pareil en ce qui concerne les fils électriques qui étaient vraiment modélisés en détail dans la scène d'origine. J'ai remplacé ces objets complexes par de simples géométries plates contenant une texture de fils avec un canal d'opacité. La 3D, c'est souvent des décors de théâtre, comme le résume très bien David Rousset.

Mais le gros du travail a porté sur les colonnes de l'Hôtel de ville, puisque celles-ci contenaient plus de 10 000 triangles, ce qui, en plus d'être trop important pour la scène dans son ensemble, déséquilibrait beaucoup trop le rendu, provoquant un gros « FPS drop », une chute du nombre d'images à la seconde, lorsque ces objets passaient dans le champ de vision. Pour ceci, les logiciels de création modernes permettent souvent de calculer dans une texture l'aspect d'un objet complexe sur une géométrie « proxy » bien moins détaillée, technique qui a été employée ici pour les colonnes et sur l'intérieur de la voiture (en groupant les géométries, et en les simplifiant), avec une belle remontée des performances à la clef.

Bien sûr, on fait une passe sur les géométries pour faire la chasse aux faces inutiles (celles qui ne seront jamais vues du point de vue de la caméra, ou trop petites, ou non pertinentes). En plus de booster les performances, cela a un impact direct sur la taille du fichier descriptif.

Et maintenant, on rentre ?

Non, pas encore. On pourra aussi porter son attention sur tout ce qui peut faire perdre des images par secondes dans un contexte particulier. Par exemple, et pour simplifier de manière un peu radicale, les moteurs WebGL pour navigateurs sont mono-threadés, à cause de certaines limitations de javascript c'est-à-dire qu'on ne peut pas beaucoup paralléliser le traitement de rendu. Donc on veillera à éviter certaines options de rendu un peu gourmandes en terme de « draw calls » (itérations de rendu pour arriver à l'image finale), comme les miroirs temps-réel ou les matériaux mixés (deux matériaux différents qui vont se pondérer par l'intermédiaire d'une texture en noir et blanc). Tout ce qui ralenti le rendu doit disparaître !

Pour conclure

Vous l'avez compris, la 3D sur navigateur en version mobile recèle quelques embûches pour quiconque penserait pouvoir simplement convertir un existant. Une phase d'optimisation sera indispensable en attendant une inévitable augmentation des performances de nos appareils, et une amélioration des APIs et des méthodes existantes. En attendant, vous serez bientôt en mesure de contempler la scène Hill Valley sur votre Windows Phone favori, puisque la compatibilité WebGL devrait arriver avec la version de l'OS 8.1, disponible très bientôt ! Enfin et pour mémoire, la totalité de l'archive de cette scène 3D est passée de 350 Mo à 57 Mo, le fichier descriptif a maigri de 41 Mo à 17 Mo, le nombre de fichiers de texture de 438 à 142 et le framerate (nombre d'image par seconde) sur une plateforme donnée de 3 ips à plus de 60 avec les méthodes décrites ici. Opération rentable, donc.

Michel Rousseau
Évangéliste technique Ux, Ui et desgn
Microsoft France

Sous le capot d'un moteur 3D : Babylon.js

J'ai commencé à développer à l'âge de 9 ans. Mes parents pour me motiver à faire ma communion m'avaient dit que je recevrais plein de cadeaux et d'argent de la part des invités. Ni une ni deux, je me suis dit que cela valait le coup et je me suis donc acheté un Amstrad 464 avec lequel j'ai clairement commencé à apprendre ce qui plus tard deviendrait mon métier.

Je n'ai aucun talent pour l'art ou la musique, même si j'adorerais savoir dessiner et je pense que le développement fut ma manière d'exprimer une créativité débordante. Cette créativité, j'ai eu très tôt envie de la mettre en œuvre pour créer des mondes (Le syndrome de l'aquarium sans doute). A la manière dont je pouvais passer des heures à regarder des fourmis travailler dans une fourmilière j'ai eu envie de créer des mondes et de les voir bouger et évoluer. C'est pour cette raison que je me suis lancé dans la création de moteurs 3D. Les premiers furent développés en Pascal, puis rapidement en C puis C++ (dont un nommé **z3d** pour d'obscures raisons que j'utilisais lors de demo-parties, comme par exemple celle-ci : https://www.youtube.com/watch?v= y13HrDoBBps). Lorsque .NET pointa le bout de son nez, j'ai créé, avec l'aide de mon ami Michel Rousseau, le moteur qui nous fit vivre pendant plus de 11 ans. Ecrit en C#, **Nova** fut le centre de mes attentions pendant de longues années. Ce n'est qu'à regret que j'ai dû le laisser tomber pour venir travailler chez Microsoft.

Là, seulement quelques jours après mon arrivée j'ai découvert que nous allions sortir une version de Silverlight avec le support de la 3D. Ni une, ni deux, me voilà parti pour créer un nouveau moteur nommé **Babylon**. Vous pouvez encore en trouver la trace ici : http://code.msdn.microsoft.com/Babylon-3D-engine-f0404ace. Lorsque IE11 sortit avec le support de WebGL, je compris que c'était un message de l'univers : je devais continuer à faire des moteurs 3D, mais pour le web en open source. Ainsi naquit babylon.js que vous pouvez trouver ici : www.babylonjs.com. Développé avec l'aide de David Rousset, Pierre Lagarde et Michel Rousseau, ce moteur occupe actuellement 100% de mon temps libre et je vais vous conter ici comment babylon.js est architecturé en tentant de vous donner également le contexte qui a amené aux différents choix.

KISS: Keep it simple and smart

La première règle qui gouverne tous les choix que nous faisons dans Babylon.js est la simplicité d'utilisation. Ce choix assumé par l'équipe impose que l'API de haut niveau, c'est-à-dire celle utilisée par les développeurs soit la plus simple possible. Moins de ligne de code il y a, mieux c'est; « the lesser the better » comme disent mes nouveaux collègues.

Cela peut avoir certaines conséquences dans nos choix que nous verrons plus bas. Par exemple, pour nos cameras nous aurions pu opter pour une caméra générique qui prend en paramètres des comportements genre « gère le tactile » ou « gère les collisions ». Architecturalement cela aurait de la gueule mais pour l'utilisateur lambda cela rajoute une complexité inutile. Nous avons donc opté pour des caméras spécialisées quitte à ce qu'en interne, elles utilisent le mécanisme décrit plus haut.

Les contraintes

Nos contraintes principales étaient les suivantes :

- Maximiser les performances : ce point va piloter la majeure partie de l'architecture interne comme nous allons le voir ci-après,
- S'intégrer à la manière de travailler des graphistes : ce point implique que notre modèle objet soit dicté par les objets manipulés par les graphistes,
- Garder une API de haut niveau simple et directe.

Les entités

Les graphistes ont l'habitude de travailler avec des scènes, des meshes, des lumières, des caméras et des matériaux. Nous avons donc tout naturellement gardé ces mêmes entités dans notre modèle objet.

Voici donc les acteurs de notre moteur 3D :

- **Engine** : Cet objet central est le responsable de la liaison avec WebGL. Il est instancié une fois pour toute et sert à masquer la complexité inhérente à une API de bas niveau comme WebGL
- **Scène** : il s'agit du container global dans lequel sont ajoutées toutes nos entités,
- **Camera** : une caméra définit le point de vue de l'utilisateur et se charge de son interaction en prenant en charge les entrées / sorties (tactile, souris, clavier, device orientation, Oculus, gamepad),
- **Meshes** : ce sont les objets 3D à dessiner,
- **Matériaux** : un matériau définit comment dessiner un mesh (en gros la couleur de chacun de ses pixels),
- **Lumières** : elles sont utilisées en conjonction avec les matériaux pour définir la couleur des pixels.

Garantir la performance coûte que coûte

Pour être efficace un moteur 3D doit envoyer la patate douce comme on dit par chez moi. En gros il doit afficher le plus d'images possible par seconde en essayant de garder une fréquence de 60 trames par seconde garantissant une fluidité parfaite.

Pour ce faire, voici l'algorithme général pour dessiner une image:

- En fonction du point de vue de la camera, sélectionner les objets visibles,
- Pour chaque objet :
  - Activer son matériau,
  - Transmettre les informations du matériau a WebGL,
  - Transmettre les informations de l'objet a WebGL,
  - Donner l'ordre de dessin de l'objet.

Nous allons voir que chaque étape nécessite son lot d'optimisations.

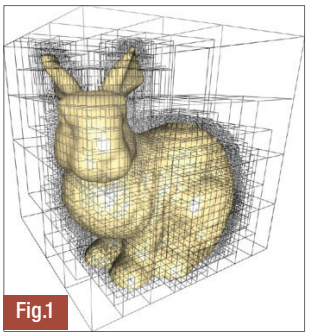


Fig.1

Un octree en action (Image fournie par Nvidia)

Sélection des objets à dessiner

Pour aller le plus vite possible, l'une des stratégies consiste à réduire au maximum le nombre d'objets envoyés à la carte graphique.

Pour ce faire, babylon.js s'appuie sur plusieurs techniques :

- **Octree** : lorsque le nombre d'objets dans une scène est suffisant (disons au-dessus de 100 objets), babylon.js va construire un octree. Cette structure de données permet de découper l'espace de la scène en 8 parties (en 8 boîtes, cf. figure 1). Babylon.js va ensuite ranger chacun des objets dans une de ces boîtes. L'objectif est ici de faire une première sélection rapide des objets visibles. Si la camera ne voit pas une boîte, nous n'avons pas besoin de vérifier tous les objets qu'elle contient. Ainsi, si on considère une scène avec 64 objets équitablement répartis dans l'espace, chaque boite contiendra 8 objets. Et donc, au lieu de tester si la camera voit chacun des 64 objets (nous allons voir après comment) elle va faire 8 tests sur l'octree et va définir qu'elle ne voit par exemple que 3 boîtes et donc fera son test de visibilité sur 24 objets. Donc 24 + 8 = 32 tests au lieu de 64. Evidemment plus il y a d'objets plus

les octrees sont efficaces. De plus ils peuvent être récursifs (chaque boîte peut à nouveau contenir un octree)

- Vient ensuite ce que l'on appelle le **frustum clipping** ou la pyramide de vision. La camera va vérifier si les objets sont ou non dans son champ de vision afin de dresser une liste d'objets visibles (voir figure 2),
- Lorsque ces objets sont très gros (genre 30000 faces ou plus), ils sont eux-mêmes découpés en sous-objets qui vont suivre le même chemin : octree (comme sur la figure 1) puis frustum clipping.

On se retrouve donc avec la liste la plus petite possible d'objets à dessiner à l'écran. Si les objets ne sont pas transparents, ils sont triés du plus proche au plus lointain, afin de profiter du depth buffer. Ce dernier fonctionne ainsi : à chaque fois qu'un pixel doit être dessiné à une certaine coordonnée (x, y : en 2d puisque nous parlons de pixels sur l'écran), le depth buffer (ou tableau de profondeur) vérifie qu'il n'y a pas un pixel déjà écrit à cet endroit mais dont la profondeur associée serait moindre (et donc plus proche de nous). Du coup en triant les objets du plus proche au moins proche on profite pleinement du depth buffer qui va éliminer beaucoup de pixels et donc réduire le travail du GPU. Toutefois, pour les objets transparents cela ne fonctionne plus car pour faire de la transparence il faut que quand un pixel se dessine, le précèdent soit déjà là (pour justement le voir en transparence).

Optimisation des shaders

Une fois que les objets à dessiner sont listés et rangés, Babylon.js va les rassembler par shaders afin d'éviter des changements de contexte dans WebGL. En effet, WebGL est très sensible aux changements d'états, et dans la mesure du possible, il faut essayer de minimiser ces changements. Ces shaders sont produits par babylon.js à partir d'un gros shader global capable de réaliser toutes les opérations supportées. Ce shader est compilé à la volée par Babylon.js en fonction du matériau de chaque objet. Par exemple, tel objet n'aura pas besoin du spéculaire et donc Babylon.js partira du shader global grâce à des « #define » présents dans le code. Le shader global ressemble donc à ceci :

```
  // Bump
  vec3 normalW = vNormalW;

#ifdef BUMP
  normalW = perturbNormal(viewDirectionW);
#endif

  // Ambient color
  vec3 baseAmbientColor = vec3(1., 1., 1.);

#ifdef AMBIENT
  baseAmbientColor = texture2D(ambientSampler, vAmbientUV).rgb * vAmbientInfos.y;
#endif
```

Ainsi dans cet exemple, pour les matériaux qui n'utilisent pas le bump, le shader sera compilé sans le #BUMP. Ainsi chaque objet sera dessiné par un shader parfaitement adapté à son matériau et ceci sans aucun IF dans le shader. En effet, il est important de noter que les GPU sont très puissants pour exécuter du code en parallèle mais détestent les tests qui cassent leur pipeline en rajoutant des « jump » dans leur flot. Bien évidemment un cache de shaders déjà compilés est présent dans Babylon.js afin de réduire au maximum le nombre total de shaders différents car nous souhaitons maximiser la réutilisation.

Cache

Toujours dans cet esprit de réutilisation et en gardant à l'esprit que WebGL déteste les changements de contexte, babylon.js propose un système de cache a plusieurs niveaux pour chaque fonctionnalité de WebGL.

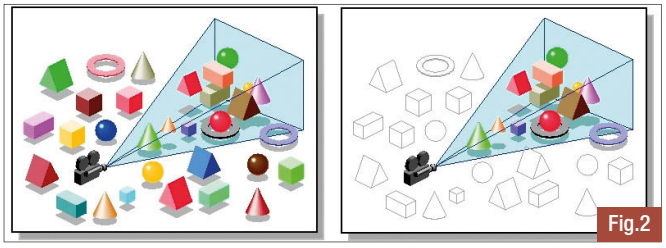


Fig.2

Frustum clipping en action (Image fournie par SlimDX)

Ainsi les ordres ne sont effectivement envoyés à WebGL que s'il y a un changement effectif. De plus les gros calculs de matrices effectués sur les objets ne sont également réalisés que si quelque chose change. Ainsi une scène statique comme Espilit (www.babylonjs.com/?ESPILIT) ne coûte presque rien en CPU car toutes les matrices sont stockées et réutilisées vu qu'aucun objet ne bouge.

Réduire la pression sur le garbage collector

Finalement un des gros axes de travail en plus du cache sur Babylon.js s'est situé au niveau de la réutilisation maximale des structures de données. Comme de nombreux langages, JavaScript possède un garbage collector (ou ramasse-miettes) qui prend soin de la mémoire pour nous. Mais comme toujours, ce service n'est pas gratuit et peut même devenir gênant dans le cadre d'un moteur 3D où la moindre baisse de vitesse de rendu donne une désagréable sensation de saccade. Or si l'on n'y prête pas attention, il est facile de faire trop travailler le garbage collector. Si l'on reprend la scène Espilit, sans aucune optimisation, elle va générer 6000 matrices (donc 16 valeurs flottantes) par image, et donc, 60 fois par seconde. Cela va vite entraîner un appel au garbage collector pour nettoyer la mémoire. Nous avons donc dû passer pas mal de temps à revoir nos structures pour que chaque fois que possible les opérations réutilisent des structures déjà créées au lieu d'en générer de nouvelles. Par exemple voilà comment était codée la somme de deux vecteurs avant :

```
public add(otherVector: Vector3): Vector3 {
    return new Vector3(this.x + otherVector.x, this.y + otherVector.y, this.z + otherVector.z);
}
```

Et maintenant :

```
public addToRef(otherVector: Vector3, result: Vector3): void {
    result.x = this.x + otherVector.x;
    result.y = this.y + otherVector.y;
    result.z = this.z + otherVector.z;
}
```

La différence se situe bien dans le fait que plutôt que de faire un new, nous stockons le résultat dans une variable passée en paramètre. On perd en beauté d'écriture mais on gagne fortement en performances (comme souvent).

Et ce n'est pas tout

Tout le plaisir de développer un moteur 3D réside de mon point de vue dans la guerre constante contre la vitesse de rendu. Chaque nouvelle fonctionnalité apporte un risque, et doit donc être bien intégrée et pensée pour ne pas déséquilibrer le fragile édifice. J'aime énormément chasser les goulets d'étranglements avec parfois des résultats décevants. Je ne compte plus le nombre de soi-disant optimisations qui font perdre de la performanceJ. Par contre, quand une d'entre-elles marche c'est une vraie victoire contre la machine. Je pourrai écrire des pages entières sur toutes les autres optimisations que nous avons apportées à Babylon.js (Instances, niveau de détail, batch de textures, textures compressées, etc..). Dans un premier temps je vous invite à vous connecter sur www.babylonjs.com et à télécharger le code source. Si cela vous plait, n'hésitez pas à contribuer, on adore ça !

David Catuhe - *@deltakosh*
Senior Program Manager - IE /Open Web Standards - Microsoft

Soyez libres ! Choisissez votre SDK de Reporting !

Il faut le dire! Ça coûte souvent cher de réaliser sur mesure un logiciel ou progiciel répondant à ses besoins précis. De même quand vous retrouvez plus de 60% de vos besoins au sein d'un logiciel du marché, vous avez bien envie de l'utiliser ! Seulement vous voulez qu'il puisse couvrir les 40% restants de vos besoins. Les logiciels de reporting n'échappent pas à cette règle.

Combien de développeurs de rapports se sont vus demander par leur entreprise de transformer leurs rapports en factures. Bien évidemment, un rapport n'est pas une facture. Mais grâce à l'extension d'un moteur de rapport, on peut bien effectuer cette transformation. Et tout cela grâce au SDK du serveur de rapport.

SDK Reporting, une offre d'indépendance !

Il arrive, au moins une fois dans un projet de développement de rapports, qu'on vous dise qu'on veut améliorer l'interface graphique, ou qu'on veut pouvoir agir sur le mode de distribution des rapports. Et dans la majeure partie des cas, l'outil de reporting par défaut ne vous le propose pas. Alors même si vous êtes un bon développeur de rapports BI, vous allez devoir mettre la main à la pâte.

Et c'est là que le SDK entre en jeu. Le SDK de reporting est malheureusement très mal connu, mais rend bien service en offrant une certaine liberté (Fig.1). D'ailleurs, une SSII dont je ne citerai pas le nom, a pu développer une solution capable de migrer un univers Business Objects en cube Microsoft Analysis Services. Et cela grâce à l'utilisation de 2 SDK (SAP Business Objects et Microsoft Analysis Services). Maintenant essayons de voir un cas pratique.

Business Objects : le module Webi

Business Objects est une solution de business Intelligence dont le module Webi (ou Web Intelligence pour les moins familiers) est le plus connu pour son aspect business intelligence « analytique ». C'est grâce à lui que les décideurs peuvent faire parler leurs données (en faisant des croisements, des comparatifs ou des forages de données). Ce module comme tous les autres, peut être customisé à volonté. Il faut juste connaître et utiliser le bon SDK. Dans notre cas, nous allons voir comment utiliser le SDK Report Engine qui permet de customiser Webi.

Préparation de l'IDE et gestion des bibliothèques

La première étape consiste à choisir son IDE. Bien sûr le choix de l'IDE dépend du SDK. En effet, deux SDK peuvent être utilisés : .NET et Java. Les partisans de .NET utiliseront certainement Visual Studio alors que les codeurs Java auront l'embarras du choix (personnellement j'utilise Eclipse comme IDE). L'autre considération concerne le type d'application. Webi étant un module permettant une navigation dans les données via un browser, il se présente sous une forme web et nécessite donc l'installation d'un projet d'application web (sous Eclipse, WTP est un bon choix). L'étape suivante consiste à se documenter sur les APIs qu'on va devoir utiliser. La documentation de ces APIs est disponible sur le site de SAP où on trouve la javadoc entière du Report Engine.

La dernière étape avant le développement, consiste à récupérer les librairies (dans notre cas les fichiers .jar du Report Engine) et les déclarer dans le projet à mettre en place.

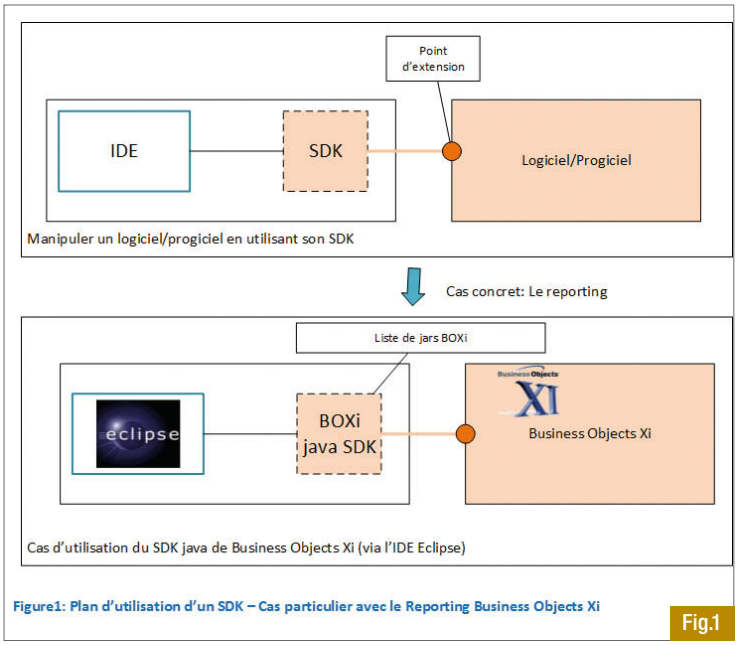


Figure1: Plan d'utilisation d'un SDK – Cas particulier avec le Reporting Business Objects Xi

Fig.1

Description du use case

Webi est généralement utilisé pour faire un bon nombre de choses dont :

- La création d'un nouveau document d'analyse,
- Le forage dans un document,
- Le rafraîchissement d'un document.

Dans notre cas, nous allons juste décrire comment créer un nouveau document, y rajouter des données puis ensuite enregistrer le document dans le référentiel Business Objects.

Connexion au serveur de rapport BO

L'étape préalable à toute utilisation d'un module du serveur de rapport est la connexion. Nous allons donc nous connecter au serveur de rapport avec un login, un password et un type d'authentification. BO offre plusieurs types d'authentification (basé sur LDAP, basé sur le référentiel BO ou basé sur Windows). Le choix dépend bien évidemment du mode d'authentification choisi par votre entreprise.

```
// Connect to CMS
ISessionMgr sessionMgr = CrystalEnterprise.getSessionMgr();
IEnterpriseSession enterpriseSession = sessionMgr.logon(user, pass, host, auth);
```

Initialisation du moteur Webi

Une fois connecté au serveur Business Objects, l'étape suivante consiste à initialiser le moteur désiré parmi les multiples moteurs dont BO dispose. Dans notre cas il s'agit du moteur Webi.

```
// Initialize Webi report engine
ReportEngines reportEngines = (ReportEngines) enterpriseSession.getService("ReportEngines");
ReportEngineType type = ReportEngines.ReportEngineType.WI_REPORT_ENGINE;
ReportEngine reportEngine = (ReportEngine) reportEngines.getService(type);
```

Récupérer l'univers sur lequel le nouveau document Webi sera basé

Maintenant que le ReportEngine (ou le moteur Webi) a été initialisé, nous pouvons commencer à l'utiliser. Mais avant cela, il est nécessaire de récupérer l'univers (ou encore la couche sémantique) sur lequel il sera basé. Nous utiliserons un univers nommé *Island Resorts Marketing*.

```
IInfoStore infoStore = (IInfoStore) enterpriseSession.getService("InfoStore");
String unvQuery = "select * from CI_APPOBJECTS where SI_KIND = 'Universe'"
                + " and SI_NAME='Island Resorts Marketing'";
IInfoObjects infoObjects = (IInfoObjects) infoStore.query(unvQuery);
IInfoObject infoObject = (IInfoObject)infoObjects.get(0);
```

Mais pour récupérer un univers, comme tout objet BO d'ailleurs (Rapports, Designer, InfoView, etc.), il faut requêter sur une base de données d'information appelée Info Store. Dans les anciennes versions de BO, il fallait directement interroger la base du référentiel BO. Avec l'arrivée des versions de type Xi, on passe par un service sur lequel on peut requêter via la méthode « query ». Dans notre cas, nous requêtons sur l'info Store en récupérant dans APPOBJECTS (contenant l'ensemble des applications BO), l'univers *Island Resorts Marketing*.

Création d'un nouveau document

Nous pouvons à présent créer un nouveau document Webi en nous adressant au report Engine et en lui passant en paramètre notre univers récupéré.

```
DocumentInstance documentInstance = reportEngine.newDocument("UnivCUID="+infoObject.getCUID());
```

Récupération d'un microcube

Sous BO, les résultats des requêtes sont conservés dans des microcubes ou encore Data Provider. Ces derniers permettent de stocker des requêtes pour la restitution de données. Nous allons donc récupérer notre requêteur pour y charger *projections et conditions*.

```
DataProviders dps = documentInstance.getDataProviders();
DataProvider dataProvider = dps.getItem(0);
Query query = dataProvider.getQuery();
DataSource dataSource = dataProvider.getDataSource();
```

Création et exécution de requête

Maintenant passons à la création de la requête. Elle se fait en deux phases :

- L'ajout de la projection (ou encore « select ») à la requête,
- L'application du filtre *Pays= « US »* à la requête.

```
DataSourceObjects objects = dataSource.getClasses();
query.addResultObject(objects.getChildByName("Service"));
query.addResultObject(objects.getChildByName("Revenue"));
ConditionContainer container = query.createCondition(LogicalOperator.AND);
ConditionObject conditionObject = container.createConditionObject(objects.getChildByName("Country"));
FilterCondition filterCondition = conditionObject.createFilterCondition(Operator.EQUAL);
filterCondition.createFilterConditionConstant("US");
```

Une fois la requête passée, il faut l'exécuter pour récupérer le résultat

```
dataProvider.runQuery();
```

Formatage du rendu

Nous en arrivons à la phase la plus longue qui consiste à formater le rapport créé pour le présenter aux utilisateurs finaux. Un document contient :

- Un entête
- Un corps
- Un pied de page

A l'intérieur du corps, on peut trouver plusieurs éléments tels que les blocs et cellules. Les blocs sont sous de multiples formats : tableaux, graphiques, tableaux croisés dynamiques, etc. A tous ces éléments on peut bien évidemment appliquer des styles.

```
ReportStructure reportStructure = documentInstance.getStructure();
ReportContainer reportContainer = reportStructure.createReport("My Report"
ReportBody reportBody = reportContainer.createReportBody();
ReportBlock reportBlock = reportBody.createBlock();
ReportDictionary reportDictionary = documentInstance.getDictionary();
BlockAxis hAxis = reportBlock.getAxis(TableAxis.HORIZONTAL);
hAxis.addExpr(reportDictionary.getChildByName("Service"));
hAxis.addExpr(reportDictionary.getChildByName("Revenue"));
 SimpleTable simpleTable = (SimpleTable)reportBlock.getRepresentation();
 CellMatrix bodyMatrix = simpleTable.getBody();
 CellMatrix headerMatrix = simpleTable.getHeader(null);
 CellMatrix[] matrices = new CellMatrix[]{bodyMatrix, headerMatrix};
 for (CellMatrix matrix : matrices) {
    for (int i=0; i<matrix.getColumnCount();++i)
    {
       TableCell cell = matrix.getCell(0, i);
       Attributes attributes = cell.getAttributes();
       if (matrix == bodyMatrix) {
          attributes.setBackground(Color.white);
          attributes.setForeground(Color.black);
       } else {
          attributes.setBackground(new Color(81, 117,185));
          attributes.setForeground(Color.white);
       }
       SimpleBorder border = (SimpleBorder)attributes.getBorder();
       border.setSize(BorderSize.NONE);
       cell.setAttributes(attributes);
       Font font = cell.getFont();
       font.setName("Arial");
       font.setSize(10);
       cell.setFont(font);
       Alignment alignment = cell.getAlignment();
       alignment.setHorizontal(HAlignmentType.LEFT);
       cell.setAlignment(alignment);
       cell.setWidth(50);
    }
 }
documentInstance.applyFormat();
```

Bien sûr, j'ai simplifié les choses car, dans bien des applications, une fois le formatage effectué, il faut l'afficher dans l'application Web dédiée.

Sauvegarde du rapport

Maintenant que nous avons notre document bien formaté, il va falloir le sauvegarder et donc le mettre à disposition des utilisateurs concernés dans l'entreprise. Pour cela, on récupère un répertoire dans le serveur de rapport BO pour y déposer notre document final. Et comme précédemment, il faut interroger l'Info Store pour récupérer le nom du répertoire dans lequel nous allons enregistrer notre document.

```
String folderQuery = "select * from CI_INFOOBJECTS where SI_KIND = 'Folder'"
                   + " and SI_NAME='Report Samples'";
infoObjects = (IInfoObjects) infoStore.query(folderQuery);
infoObject = (IInfoObject)infoObjects.get(0);
int folderId = infoObject.getID();
documentInstance.saveAs("Test", folderId, null, null, true);
documentInstance.closeDocument();
```

Conclusion

Nous avons vu à travers cet article, comment utiliser à moindre coût les fonctionnalités d'un moteur de rapport. Et le grand avantage est que, nous l'avons fait librement, sans dépendre des contraintes que nous aurait imposées l'outil par défaut. Dans notre cas, nous aurions pu utiliser des fonctionnalités avancées de la Business Intelligence tels que le forage ou le pivot. De même, nous aurions pu utiliser les APIs adéquats pour transformer le même moteur en serveur d'impression de facture (Cette idée émane de *M. Chun Tah NEW* responsable d'applications de Reporting chez BNP Paribas dont je remercie au passage l'implication). Et heureusement, pratiquement tous les éditeurs de logiciels de BI offrent l'extension de leur solution via l'utilisation de leur SDK.

Mohamed **Taslimanka SYLLA**
Architecte Business Intelligence - SFEIR

Développer des applications avec AngularJS et WakandaDB

1ère partie

AngularJS est un framework Ajax. Il utilise des mécanismes de "Dépendance Injection" avec une inspiration MVC. Son « data-binding », ses « scopes », et ses « watchers » rendent le code métier plus court et plus élégant. Peu de « widgets » ici, on y retrouve une forme de magie, telle qu'on peut l'apprécier dans Wakanda, avec une approche "pure code" au cœur du HTML5.

Wakanda propose une plate-forme complète pour créer et faire tourner des applications Web : un serveur applicatif, une base NoSQL Objet, et un framework HTML5, le tout en Open Source. Elle propose également un IDE, et un service de déploiement Wakanda Cloud est dans les tiroirs. Petit bonus : Wakanda est une solution réalisée par une boîte française... Cocorico... Wakanda repose fortement sur les standards du Web, ce qui en fait un compagnon de route très agréable pour AngularJS. C'est une alternative intéressante face aux autres solutions de persistance et de partage des données.

Au cœur des données

La grande majorité des applications Web que nous utilisons tirent leur force dans leur capacité à manipuler des données. Celles-ci peuvent être locales (géolocalisation, texte saisi, luminosité, ...), mais la plupart viennent très souvent d'un serveur, via un service Cloud (Google Apps APIs, ..) ou depuis vos propres base de données, SQL ou NoSQL. Wakanda vous permet :

- de modéliser une structure de données sous forme d'objet métiers, un peu comme du UML,
- d'y inclure relations et transactions comme le feraient des RDBMS SQL classiques,
- d'y accéder en langage objet (JavaScript) depuis le serveur sans couche ORM,
- d'y accéder nativement via une API REST/HTTP depuis n'importe quel client,
- d'obtenir un miroir du modèle du serveur dans le client depuis des connecteurs dédiés.

Le connecteur qui nous intéressera aujourd'hui est bien sûr « angular-wakanda ».

Mode Coding Dojo « on »

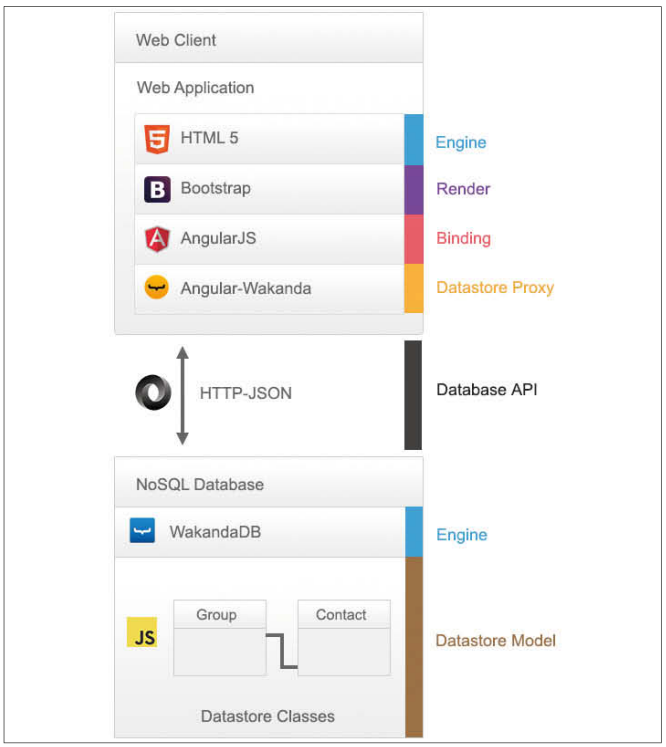
La documentation Wakanda avec ses vidéos et tutoriaux permet une prise en main rapide de la plateforme. Le « Data Model Designer », ou le « GUI Designer », du Wakanda Studio vous feront bon accueil, mais il est bon de rappeler qu'il s'agit d'outils « proposés », et que vous restez libres de préférer les vôtres. Nous allons donc opter ici pour une approche plus « angular », mais vous pouvez si vous le préférez suivre cette exercice avec Wakanda Studio. La plupart des tutoriaux angular vous inviteront à utiliser la suite d'outils Yeoman qui comprend Yo, Grunt (ou sa variante Gulp), et Bower (ou son alternative npm). Ne craignez rien si vous ne les connaissez pas, nous en ferons, pour vous initier un minimum, une utilisation restreinte et facultative.

Naissance du projet de l'application

Le projet Wakanda

La première étape est de créer un projet. L'approche Yeoman se résume de 1 à 3 instructions en ligne de commande :

```
# installe yeoman s'il n'est pas encore disponible
> npm install -g yo
# installe le générateur de project wakanda s'il n'est pas encore disponible
> npm install generator-wakanda-project
```




```
# initialise un nouveau projet
> yo wakanda-project
```

Le générateur vous demandera si vous souhaitez créer une solution Wakanda. Dites oui, celle-ci est nécessaire pour faire tourner un ou plusieurs projets sur le serveur. Wakanda Serveur peut être téléchargé depuis http://wakanda.org/downloads. Les "All-in-one Installer" pour linux sont en fait des packages debian.

Notes

Le Wakanda studio propose « new project » depuis son menu et sa barre d'outils.
Ce projet peut également être récupéré dans le NG-Wakanda-Pack disponible sur github.

Un modèle NoSQL défini sur le serveur

L'étape suivante consiste à créer un modèle définissant des objets « stockables », et c'est là toute la force de Wakanda. Créons donc un modèle pour une gestion de contacts en y intégrant 2 modules simples :

```
// Model.js
model.Contact = require('Model/Contact');
model.Group = require('Model/Group');
```

Le constructeur Attribute attend un type d'attribut (storage, calculated, alias, relatedEntity, ..), et un type de donnée qui peut être un type JavaScript ou un nom de DataClass (ex : number, string, Group).

```
// Modules/Group.js
var Group = module.exports = new DataClass();
```

```
Group.ID = new Attribute('storage', 'number', 'key auto');
Group.name = new Attribute('storage', 'string');
Group.contacts = new Attribute('relatedEntities', 'ContactCollection', 'group', {reversePath:true});
```

```
// Modules/Contact.js
var Contact = module.exports = new DataClass();
Contact.ID = new Attribute('storage', 'number', 'key auto');
Contact.name = new Attribute('storage', 'string');
Contact.mobile = new Attribute('storage', 'string');
Contact.group = new Attribute('relatedEntity', 'Group', 'Group');
Contact.groupName = new Attribute('alias', 'string', 'group.name');
```

2 DataClasses ont été définies : Group et Contact. Toutes deux ont un ID auto-alimenté permettant de les identifier. Les attributs 'storage' vont enregistrer les données (name, mobile) et les attributs relationnel 'group', permettant de retrouver le groupe auquel appartient un contact, et 'contacts', son lien retour permettant de retrouver tous les contacts associés au groupe. L'alias 'groupName' permet d'éviter les parcours manuels de relations.

Des API JavaScript Serveur et HTTP natives

Notre Modèle serveur est maintenant prêt à fournir un service de stockage et partage de données via une API REST utilisable par AngularJS. Vous pouvez le tester en lançant l'application sur le serveur en ligne de commande.

```
# lance le projet sur le server et ouvre l'URL /rest/$catalog
> grunt serve --catalog
```

Note

Le studio lance le projet via l'option « run project » de sa barre d'outils. Il ouvre par défaut la page index.html ou index.waPage/index.html en fonction de ce qui est présent.

Nous constatons dès maintenant 2 choses très intéressantes :

- La première est que Wakanda ne nécessite aucune couche ORM applicative. Aucun code supplémentaire n'est généré, et les requêtes HTTP ne sont pas converties en SQL mais correspondent au langage natif du serveur au même titre que le server-side JavaScript.
- La seconde est que ces langages de requête permettent de ne plus se soucier de jointure entre des tables. Les relations fonctionnent comme des références d'objet en JavaScript, ou des liens dans la version HTTP.

De l'obscurité à la lumière

Vous êtes pour l'instant au niveau du DaaS (Data as a Service), ce qui est déjà pas mal. Mais un utilisateur lambda ne se contentera généralement pas d'une interface en HTTP pour appeler votre modèle métier, donc affrontons dignement les arcanes du HTML5. Le projet Wakanda comprend par défaut un dossier « Webfolder ». Depuis la ligne de commande, vous pouvez aisément y charger le connecteur angular-wakanda qui récupérera automatiquement Angular s'il n'est pas déjà disponible.

```
# installe bower s'il n'est pas encore disponible
> npm install -g bower
# charge angular-wakanda et ses dépendances
> bower install angular-wakanda
```

Un dossier « bower_components » est créé avec les indispensables fichiers « angular.min.js » et « angular-wakanda.min.js ».
Nous pouvons maintenant les utiliser depuis notre page HTML.

```
<!-- Webfolder/index.html -->
<!DOCTYPE html>
<html ng-app="Contact">
<head>
  <meta charset="utf-8">
  <title>Contact Demo</title>
</head>
<body ng-controller="Controller">
  <p>angular-wakanda <strong>{{loaded}}</strong> </p>
  <script
src="/bower_components/angular/angular.min.js"></script>
  <script src="/bower_components/angular-wakanda/angular-wakanda.
min.js"></script>
  <script src="./app.js"></script>
</body>
</html>
```

Angular utilise des « directives » exprimées ici sous forme d'attributs HTML préfixés par « ng- ». « ng-app » définit la zone active de votre page et indique le nom du module requis pour démarrer l'application. « ng-controller » établit un lien entre des sections de la page et des « scopes » du modèle JavaScript. Le scope dédié à une zone HTML est exposé en JavaScript via le paramètre $scope (voir plus bas dans app.js).
Comme « moustache.js », Angular utilise les doubles accolades pour insérer les données dynamiques dans le HTML. Ainsi, {{loaded}} représente la valeur de « $scope.loaded ».
Cette notation peut également représenter des expressions. Contrairement à la plupart des autres templates, ceux-ci sont mis à jour en temps réel dès qu'une propriété de $scope change de valeur.

```
// Webfolder/app.js
angular.module('Contact', ['wakanda']);
function Controller($scope, $wakanda) {
    $scope.loaded = 'loading';
    // Create a proxy of the server model
    $wakanda.init().then(function oninit(ds) {
        $scope.loaded = '100% loaded';
    });
}
```

Côté JavaScript, on commence généralement par définir le « module » sur lequel on veut travailler. La méthode angular.module() donne son nom et liste les dépendances nécessaires pour le code courant. Ainsi, notre module « app », demandé par la directive « ng-app », requiert l'initialisation du module « wakanda ». La fonction « Controller() » désignée par « ng-controller » reçoit le fameux paramètre injecté « $scope », et « $wakanda » qui va préparer un proxy du modèle défini sur le serveur. AngularJS bénéficie d'une API cliente dédiée avec nos DataClasses, et leurs propriétés, méthodes, et relations. Vous devez dès maintenant pouvoir observer dans le navigateur votre 1ère page angular-wakanda qui affichant : « angular-wakanda 100% loaded »

Prochain numéro

Nous avons créé une application avec un modèle Wakanda fonctionnel exposé en HTTP puis créé une interface Angular connectée et prête à l'utiliser. Nous allons maintenant pouvoir, dans le prochain numéro, créer de manière très naturelle l'interface graphique de notre application.

Ressources

https://angularjs.org
http://wakanda.org
http://yeoman.io
https://github.com/Wakanda/NG-Wakanda-Pack

Alexandre **Morgaut** - Community Manager chez 4D
http://github.com/AMorgaut

La programmation fonctionnelle en F# #2

Quand on parle de programmation fonctionnelle entre développeurs, il n'est pas rare d'entendre des commentaires du type : « j'en ai fait un peu à l'école » ou « c'est un langage pour faire du calcul scientifique, n'est-ce pas ? ». C'est malheureusement cette perception parfois un peu limitée sur la programmation fonctionnelle qui empêche ce paradigme de se positionner au même niveau que d'autres paradigmes de programmation plus courants tels que la programmation impérative ou orientée objet.

LA PHILOSOPHIE DERRIÈRE LA PROGRAMMATION FONCTIONNELLE

Tout paradigme de programmation propose une manière de résoudre des problèmes : la programmation procédurale décrit pas à pas des instructions à exécuter (C, Pascal); le paradigme objet encapsule la structure et le comportement dans des objets (C#, Java); SQL propose une programmation relationnelle et ensembliste. La programmation fonctionnelle prône une approche déclarative, nous écrivons ce que l'on veut faire et non comment le faire. Nous n'écrivons pas une série d'instructions à exécuter mais nous décrivons plutôt l'intention de ce que l'on veut faire. Les fonctions nous servent à décrire des comportements et des transformations de données. Contrairement à la programmation orientée objet où les comportements sont encapsulés dans des objets, en programmation fonctionnelle nous déclarons des structures des données sans comportement (ou presque), et nous ajoutons des fonctions pour ajouter des fonctionnalités.

LES PRINCIPAUX ÉLÉMENTS DES LANGAGES FONCTIONNELS

Les types de données

F# dispose des types primitifs (entier, flottant, booléen) et les types .NET « de base » (tableaux, dates, chaines de caractères) mais introduit également d'autres structures de données.

Les tuples

Les tuples sont des groupes de données ordonnées qui peuvent être de différents types, ce sont des structures assez simples.

```
let eqDoge = doge = ("Chien", "Doge", 5)
//val doge : string * string * int = ("Chien", "Doge", 5)
```

Tuple avec deux chaînes de caractères et un entier.

Les enregistrements (Records)

Les *records* représentent des structures de données dont les valeurs sont nommées.

```
type Animal = { Espece : string ; Nom : string ; Age : int }
let doge = { Espece = "Chien"; Nom = "Doge"; Age = 5}
```

Les unions discriminantes (Discriminated Unions)

Elles nous permettent de différentier des données parmi une classification.

```
type animalType = |Chien |Chat |Autre //discriminated union
type animal = {Espece : animalType; Nom : string; Age : int }
 //record
let doge = {Espece = Chien; Nom = "Doge"; Age = 5}
```

Les valeurs des unions discriminantes peuvent avoir des types et même faire référence à elles-mêmes. De cette façon il est extrêmement simple de créer des structures de données complexes.

```
type Arbre =
        | Feuille of int
        | Noeud of int * Arbre * Arbre
let monArbre = Noeud(1,
                     Noeud(2,
                             Noeud(4,
                                 Feuille(6),
                                 Feuille(7)),
                             Feuille(4)),
                     Feuille(3))
```

Les listes et les séquences

En F# on a des listes dites fonctionnelles qui sont en réalité des listes chaînées.

```
let premiers = [3; 5; 7]
//val premiers : int list = [3; 5; 7]

let premiersEtUn = 1::premiers
//val premiersEtUn : int list = [1; 3; 5; 7]
```

L'opérateur :: nous permet de créer une liste à partir d'un élément de tête (head) et un élément de queue (tail) qui est lui-même une liste. C'est le principe des listes chaînes.
Les séquences en F# sont évaluées de manière tardive (lazy), c'est-à-dire, qu'elles ne sont pas énumérées tant que nous ne le demandons pas – en C# nous utilisons l'interface IEnumerable.

```
let entiersPairs =
            seq{
                for i in [1..2..10] do
                    yield i
            }
//val entiersPairs : seq<int>
```

Les Classes, interfaces, structures, énumérations, delegates

Ce sont des types que nous rencontrons également dans F# mais que nous ne détaillerons pas ici car ce sont des concepts que nous trouvons également dans d'autres paradigmes de programmation comme la programmation orientée objet.

Les Fonctions

Comme vous l'avez sûrement imaginé, les fonctions sont au cœur de la programmation fonctionnelle. La syntaxe est ici relativement simple :

```
let auCarre x = x * x
//val auCarre : x:int -> int
```

La fonction *auCarre* élève *x* au carré. Vous pouvez distinguer la syntaxe presque mathématique.

La Composition de fonctions

La composition entre fonctions est semblable à la notion mathématique. Le principe est d'utiliser le résultat d'une fonction comme entrée d'une autre fonction.

```
let estPair y = y % 2 = 0
//val estPair : y:int -> bool
```

La fonction *estPair* indique si l'entier *y* est pair.
L'opérateur de composition en F# est >>.

```
let leCarreEstPair = auCarre >> estPair
//val leCarreEstPair : (int -> bool)
```

La fonction *leCarreEstPair* est maintenant une nouvelle fonction qui prend un entier en entrée, l'élève au carré (fonction *auCarre*) et indique s'il est pair (fonction *estPair*).

Le Pipelining

Le principe du pipelining est de passer le résultat d'une expression comme paramètre à une fonction. C'est un opérateur que vous rencontrerez très souvent en F#. Mais quel intérêt ? Cela permet d'enchaîner des traitements et ainsi d'éviter les appels imbriqués aux fonctions, ce qui améliore grandement la lisibilité d'un programme. Comparons ainsi :

```
let leCarreEstPair12 = estPair(auCarre(12))
let leCarreEstPair12' = 12 |> auCarre |> estPair
```

L'opérateur pipeline est vraiment appréciable quand on a un grand nombre d'appels de fonctions les uns après les autres. C'est semblable aux méthodes d'extension en C#.

Application partielle (ou curryfication[1])

C'est une notion très importante et utile en programmation fonctionnelle. Évitons une définition trop théorique et voyons plutôt un exemple. Nous avons ici une fonction prenant deux paramètres en entrée, grâce à la notion d'application partielle nous pouvons 'fixer' la valeur d'un paramètre pour créer une nouvelle fonction.

```
// le format "%+d" ajoute le signe + ou -
printfn "%+d" -3
printfn "%+d" 9
// affiche : -3 +9
```

Au lieu de répéter à chaque fois le format pour l'affichage, on peut créer une fonction en fixant la valeur du format.

```
let afficheEntier = printf "%+d "
//val afficheEntier : (int -> unit)
```

Si on prête attention à la signature de la fonction *afficheEntier* nous remarquons qu'elle reçoit un entier en entrée et qu'elle renvoie *unit* (correspond à *void*). Nous avons ainsi « transformé » une fonction qui reçoit deux paramètres en une fonction qui n'en reçoit qu'un en fixant la valeur du premier paramètre.

```
afficheEntier -3
afficheEntier 9
// affiche : -3 +9
```

La surcharge d'opérateurs

Comme en C#, il est possible de surcharger des opérateurs (+, - par exemple). Nous pouvons aller encore plus loin en F# et créer nos propres opérateurs. Créons par exemple un opérateur qui soustrait à un chiffre un pourcentage donné.

```
let (|%) valeur pourcentage = valeur * (1.0 - pourcentage/100.0)
//val ( |% ) : valeur:float -> pourcentage:float -> float
```

(1) Même s'il y a une différence entre application partielle et curryfication, elle est assez subtile et nous considérerons ces deux notions comme similaires.

Aussi simple que de déclarer une nouvelle fonction, certains opérateurs peuvent être utilisés entant qu'opérateurs *infixes*, c'est-à-dire, que l'opérateur peut être placé entre ses deux paramètres – comme le signe + ou -, qui sont d'ailleurs aussi des fonctions.

```
let prixArticle = 50.99
let remise = 80.0 // dernière démarque
let prixEnSolde = prixArticle |% remise
//val prixEnSolde : float = 10.198
```

Le pattern matching

Le terme de Pattern Matching est le principe de comparer des données par rapport à une structure (modèle/pattern) particulière.
Voyons cela avec des exemples.
Reprenons les structures de données employées précédemment :

```
let doge = {Espece = Chien; Nom = "Doge"; Age = 5}
let lol = {Espece = Chat; Nom = "Lol"; Age = 2}
let flipper = {Espece = Autre; Nom = "Flipper"; Age = 15}
let caractereAnimal animal =
    match animal with
    | {Espece = Chien} -> "Sympa"
    | {Espece = Chat} -> "Mignon"
    | {Espece = Autre} -> "Indéfini"
//val caractereAnimal : animal:animal -> string

let caractereDoge = caractereAnimal doge
let caractereGrumpy = caractereAnimal grumpy
let caractereFlipper = caractereAnimal flipper

//val caractereDoge : string = "Sympa"
//val caractereGrumpy : string = "Mignon"
//val caractereFlipper : string = "Indéfini"
```

Le constructeur '*match with*' ressemble au bien connu '*swith case*' dans d'autres langages mais plus puissant encore, par exemple, nous aurons un 'warning' si nous ne couvrons pas tous les cas. Dans l'exemple plus haut, nous serons prévenus si nous ajoutons un nouvel élément à notre union discriminante *animalType* et nous ne mettons pas à jour notre fonction *caractereAnimal*. Un autre exemple de Pattern Matching est la possibilité de décomposer des tuples en plusieurs variables :

```
let villeTuple = ("San Francisco", 800000, 600)
let (nomVille, habitants, superficie) = villeTuple
//val villeTuple : string * int * int = ("San Francisco", 800000, 600)
//val superficie : int = 600
//val nomVille : string = "San Francisco"
//val habitants : int = 800000
```

Type providers

Le principe des 'type providers' est de fournir un accès simple et sûr (en terme de typage) à des sources de données externes sans avoir à générer du code au préalable (pré compilation). Il existe une grande variété de 'type providers' : SQL, XML, CSV, JSON, OData, VMI, Hadoop/Hive, R, WSDL et bien d'autres encore.

```
type LangagesType = XmlProvider<"""<langage nom="un langage"
paradigme="un paradigme"/>""">
let langage = LangagesType.Parse("""<langage nom="F#" paradigme
="Fonctionnel"/>""")
printf "%s est un langage %s" langage.Nom langage.Paradigme
```

LangagesType définit un type par rapport à une représentation XML – qui peut être un fichier également. Ensuite il est possible de charger un conte-

nu XML correspondant à la représentation définie par *LangagesType*. On peut ensuite accéder directement aux éléments de notre fichier XML sans avoir à générer du code ni à compiler notre projet.

Unités de mesure

F# nous permet d'associer des unités de mesure à des types numériques. Cela permet au compilateur de valider que les relations arithmétiques ont les bonnes unités. Nous déclarons ainsi deux types d'unités personnalisées :

```
[<Measure>] type Habitant
[<Measure>] type Km
```

Si nous prenons le tuple *villeTuple :*

```
let villeTuple = ("San Francisco", 800000<Habitant>, 600<Km^2>)
let (nomVille, habitants, superficie) = villeTuple
let densite = habitants/superficie
//val densite : int<Habitant/Km ^ 2> = 1333
```

Par contre, nous ne pouvons pas additionner des habitants à une superficie car il s'agit d'un calcul qui n'a pas de sens (le code ci-dessous ne compile pas) :

```
//ne compile pas
let mauvaisCalcul = habitants + superficie
```

Computation expressions (monades)

Les monades nous permettent d'encapsuler des traitements et des calculs afin de pouvoir les enchaîner de manière sûre et sans effets de bord. Elles sont concrétisées par ce qu'on appelle des expressions de calcul en F#. Même si ce terme peut ne pas paraître familier aux développeurs C#, vous avez sûrement déjà manipulé des concepts semblables sans vous en rendre compte – une partie de LINQ et les opérations sur les Tasks par exemple.

Workflows asynchrones

Un exemple, est le constructeur *async* qui nous permet d'encapsuler et enchaîner des traitements asynchrones :

```
let telecharger url =
    async{
        do! Async.Sleep 1000 // asynchrone
        let webClient = new WebClient() // synchrone
        let! html = webClient.AsyncDownloadString(new Uri(url))
 // asynchrone
        return html
    }
let sites = ["http://fsharp.org";
             "http://www.tryfsharp.org";
             "https://github.com/fsharp/"]

let superContenu =
        sites
        |> List.map telecharger
        |> Async.Parallel
        |> Async.RunSynchronously
```

À l'intérieur de l'expression *async*, l'opérateur *!*(bang) permet d'extraire le résultat d'une autre opération asynchrone (précédé par *let* ou *do* dans l'exemple). De cette manière il est très simple de combiner des traitements asynchrones avec des calculs qui ne le sont pas.
superContenu contient donc le contenu html téléchargé de manière parallèle des différents sites déclarés.

Workflows de séquence

Une autre expression de calcul très utilisée en F# est le constructeur *seq* que nous avons déjà vu précédemment.

```
let unAcinq = seq { for i in 1..5 do yield i }
let sixAdix = seq{ for i in 6..10 do yield i }

let unAdix =
        seq{
            yield! unAcinq
            yield! sixAdix
        }
```

Nous créons une première séquence qui va de 1 à 5 et une deuxième de 6 à 10. Nous combinons ensuite ces deux séquences afin de créer une nouvelle séquence qui ira de 1 à 10. De manière analogue à *async*, `l'opérateur` *!*(bang) modifie le comportement du mot clé *yield* et nous permet d'extraire le contenu de chaque séquence – de l'aplatir –, comme le ferait la méthode *SelectMany* en C#.
Il existe d'autres expressions de calcul dans F# et il est même possible de créer ses propres expressions.

L'INFLUENCE DE LA PROGRAMMATION FONCTIONNELLE SUR LA PROGRAMMATION ORIENTÉE OBJET

Si vous développez dans un langage orienté objet (comme C# ou Java) vous avez sûrement remarqué que certains éléments abordés dans l'exploration du langage F# se retrouvent également dans d'autres langages. Il faut dire que la programmation fonctionnelle a joué un rôle dans l'évolution des langages orientés objet.

L'inférence de type

Comme vous avez pu remarquer dans les exemples précédents, nous n'avons déclaré aucun type explicitement – même si cela est possible et nécessaire dans certains cas –, pourtant F# est un langage statiquement typé. C'est grâce au moteur d'inférence de type de F# que cela est possible. L'inférence de type existe dans d'autres langages comme C#. Lorsque vous déclarez une variable avec le mot clé *var* ou que vous appelez une méthode générique, le compilateur essaye d'inférer le type utilisé grâce à des informations contextuelles.
Le moteur d'inférence de F# va encore plus loin car il nous permet de ne pas déclarer de type dans des contextes où en C# il serait nécessaire de le faire, comme par exemple dans les paramètres des fonctions ou le retour des fonctions – l'utilisation du mot clé *var* n'est permise que dans la portée d'une méthode. C'est comme si en C# on utilisait le mot clé *var* pour les types des paramètres des méthodes ou le type de retour d'une méthode, tout en gardant la sécurité d'un typage statique.

Les types génériques

F# (comme d'autres langages fonctionnels) traite les valeurs de la manière la plus générique possible, grâce à un procédé connu sous le nom de généralisation automatique et au puissant moteur d'inférence de type de F#. Cela veut dire que, si par exemple le paramètre d'une fonction ne dépend pas d'un type en particulier, il sera considéré comme générique.

Les expressions lambda et les méthodes anonymes

En F#, l'utilisation de méthodes anonymes est très courante, notamment lors de l'appel à des fonctions d'ordre supérieur – fonctions qui reçoivent en paramètre ou qui retournent une autre fonction.

```
let pairs = [1..100]
            |> List.filter (fun i -> i%2 = 0 )
            |> List.map (fun i -> i.ToString() )
            |> List.reduce (fun i s -> i+","+s )
//val pairs : string = "2,4,6,8,10,12,14,16...
```

Cet exemple prend les 100 premiers entiers naturels, filtre la liste en ne gardant que les chiffres pairs, transforme (*map*) chaque chiffre en chaîne de caractère et les concatène (*reduce*) en ajoutant une virgule entre chaque élément. Il n'est pas nécessaire de déclarer une fonction pour la sélection, la projection et la réduction de notre liste d'entiers. Il suffit simplement de passer par une fonction anonyme – ou expression lambda.
C#, et plus récemment Java, ont adopté cette notion afin de rendre plus lisible le code en réduisant le nombre de méthodes, de classes et de types à déclarer.

Les tuples

Les tuples et notamment la comparaison structurelle sont largement utilisés en programmation fonctionnelle, en C# le type générique tuple est apparu à partir de la version 4 du Framework .NET.

LINQ

La manipulation de séquences – des listes et des tableaux – est une tâche très courante en programmation. Les langages orientés objets utilisent une approche impérative grâce à des constructeurs de type « boucle » (loop) comme *for*, *while* ou *foreach* tandis que les langages fonctionnels utilisent une approche déclarative en appliquant par exemple des opérations de transformation, de filtrage ou d'agrégation grâce à des fonctions.
LINQ introduit une approche plus déclarative dans la manipulation des données, et, nous les développeurs C#, sommes contents de l'avoir. LINQ nous permet de faire :

```
var pairsQ = from i in Enumerable.Range(1, 100)
             where i % 2 == 0
             select i.ToString();

var pairs = pairsQ.Aggregate((i, s) => i + "," + s);
```

Ou

```
var pairs = Enumerable.Range(1, 100)
                .Where(i => i % 2 == 0)
                .Select(i => i.ToString())
                .Aggregate((i, s) => i + "," + s);
```

Les traitements asynchrones (async await)

La programmation asynchrone (Tasks, continuations, contexte de synchronisation) se voit simplifiée avec la version 5 de C# grâce à l'apparition des mots clés *async* et *await*. Cela doit vous rappeler le constructeur *async* en F# et son comportement monadique qui nous permet très facilement d'enchaîner des traitements asynchrones.

F#, C'EST DU .NET

F# est basé sur la plateforme .NET et profite donc de la richesse du Framework .NET. Il est ainsi possible d'écrire des programmes Windows Forms, WPF, Web Forms, MVC .NET et même Windows Store Apps (Windows 8.x) en F#. Il est aussi possible d'utiliser des bibliothèques écrites en C# ou Visual Basic en F# et vice-versa.

LES OUTILS DE DÉVELOPPEMENT

Visual Studio supporte entièrement F# depuis sa version 2010. Il est donc possible de créer des projets F# comme nous créons des projets C#, C++ ou Visual Basic (entre autres). Visual Studio propose une fenêtre interactive pour l'évaluation de code F# sans avoir à compiler toute la solution. Connu sous le nom de « F# Interactive » elle utilise une boucle dite REPL (Read, Evaluate, Print, Loop) et nous permet de tester au fur et à mesure nos scripts F#. Nous avons d'autres outils et plateformes de programmation pour F# comme :

- WebSharper : projet Open Source qui nous permet de créer des applications web en JavaScript et HTML 5 en F#.
- Xamarin Studio : nouvelle version de MonoDevelop, supporte le langage F# pour les plateformes Android, OS X et iOS.
- LINQPad : le bon LINQPad supporte aussi F#.
- Et beaucoup d'extensions intéressantes dans la Visual Studio Gallery.

Il est vrai pourtant que le support de F# sur Visual Studio est moins abouti que celui de C# au niveau de la refactorisation et le support de « Designers ».

LA COMMUNAUTÉ

La communauté joue un rôle très important dans l'évolution et la diffusion du langage. Elle est très présente, très active et accueille des développeurs d'horizons très différents. Il n'est pas rare de voir des développeurs C#, Scala, Java, JavaScript, Haskell et bien d'autres lors de conférences ou événements F#. Le langage F# est Open Source et récemment l'équipe F# a commencé à accepter des contributions de la communauté au niveau du compilateur. Vous pouvez aller à la rencontre de la communauté F# sur Twitter (#fsharp), GitHub, CodePlex, Stack Overflow. Petit plus si vous n'habitez en région parisienne, il y a un groupe sur Meetup pour F# (Functional Programming in F#).

CONCLUSION

Le choix d'un langage fonctionnel comme langage de programmation peut être aussi cohérent que celui d'un langage impératif ou orienté objet, d'autant plus que l'orientation de certains langages fonctionnels comme F# ou Scala est multi-paradigme – de la même manière que des langages comme C# ou Java ont adopté des notions fonctionnelles.
Les notions d'immuabilité des valeurs et l'absence d'effets de bords font des langages fonctionnels qu'ils s'avèrent très propices à la programmation parallèle et en temps réel. Une orientation qui devient de nos jours très appréciée, voire nécessaire. La programmation fonctionnelle ne se limite pas au calcul scientifique ou à l'apprentissage académique. Les grands acteurs comme Twitter, WhatsApp, Facebook ou Microsoft l'ont bien compris, ils en font une vraie utilisation pratique.
La syntaxe succincte et déclarative des langages fonctionnels ajoute une touche d'élégance tout en gardant la sécurité d'un typage statique.

Pablo Fernandez **Duran**, *SoftFluent*
http://softfluent.com - http://blogs.softfluent.com/

Références

Web

The F# Software Foundation : http://fsharp.org/
Apprendre et tester F# en ligne : http://www.tryfsharp.org/
Sources de la librairie et le compilateur F# : https://github.com/fsharp/
Le blog de l'équipe F# : http://blogs.msdn.com/b/fsharpteam/
Comprendre la programmation fonctionnelle quand on vient de C#, Java ou Python : http://fsharpforfunandprofit.com/
Membre très actif de la communauté F# (Tomas Petricek) : http://tomasp.net/

Livres

Tomas Petricek avec Jon Skeet, **Real-World Functional Programming: With examples in F# and C#**, *Manning, 2010 (9781933988924)*
Tomas Petricek et Phillip Trelford, **F# Deep Dives**, *Manning Early Access Edition, 2012 (9781617291326)*
Robert Pickering, **Beginning F#**, *Apress, 2009 (9781430223894)*

Black Hat : Ruben Santamarta dévoile les failles de sécurité des terminaux de communication par satellite

La plus importante des conférences de sécurité de la planète regroupe trois événements mythiques en août à Las Vegas : Black Hat, B-Sides et Def Con. A elle seule Black Hat compte plus de 180 conférenciers et quelques 8000 participants, dont une majorité de hackers.

Pour les chercheurs en sécurité informatique, Black Hat est une tribune idéale pour dévoiler le fruit de leurs travaux à la communauté internationale. Dans le bruit d'une foule d'annonces sur fond sonore de machines à sous, c'est ce que Ruben Santamarta, est venu faire lors de cette dernière édition.

Ruben Santamarta

Dans la peau d'un Hacker... chercheur en sécurité

Voici plus de deux mois qu'il planche sur l'analyse des failles des terminaux de communication par satellite. Ces outils jouent un rôle prépondérant dans le système mondial des télécommunications. Ruben a 32 ans, basé à Madrid, il travaille pour le compte de IOActive et fait partie de la jeune élite de chercheurs Hackers qui fait évoluer l'industrie de la sécurité. Penser et agir comme un pirate pour mieux sécuriser les systèmes, c'est son job. Cette fois, il a évalué l'état de sécurité des terminaux SATCOM, les plus largement déployés (Inmarsat, Iridium et Thuraya).

C'est en analysant le firmware de ces terminaux qu'il a établi que des acteurs malveillants étaient en mesure de détourner l'ensemble de ces équipements. Parmi les vulnérabilités détectées figurent des backdoors (portes dérobées), des certificats codés en dur (dans le matériel) ainsi que des protocoles non documentés et non sécurisés.

Ces vulnérabilités sont extrêmement critiques puisqu'elles laissent le champ libre à d'éventuels pirates pour intercepter, manipuler ou bloquer des communications, voire dans certains cas, prendre le contrôle à distance de l'équipement physique. Lors de sa démonstration à Black Hat, Ruben a remplacé à distance le firmware d'un terminal satellite utilisé dans la marine pour envoyer des signaux de détresse, par un fond d'écran humoristique. Démontrant de fait qu'il est possible de perturber, modifier et inspecter de tels environnements de communication, d'exercer une surveillance ou de lancer des cyber-attaques.

Les infrastructures de réseaux terrestres pâtissent de limites physiques et ne peuvent tout simplement pas répondre aux besoins de certaines activités. Pour combler cette lacune et obtenir de meilleures performances, de multiples constellations de satellites ont été placées en orbite autour de la Terre. Ces réseaux ont pour fonction, entre autres, de permettre aux personnes situées dans des contrées reculées d'accéder à Internet, de faciliter la navigation en toute sécurité des navires et des avions et de fournir à l'armée et aux services de secours des liaisons de communication vitales lors de conflits armés ou de catastrophes naturelles.

Parmi les activités faisant appel aux réseaux satellitaires, on trouve notamment : l'aérospatiale, le secteur maritime, l'armée et les gouvernements, les services de secours, l'industrie (plates-formes pétrolières et gazières, centrales électriques) ou encore les médias.

Pour ses recherches Ruben Santamarta a classé les infrastructures SATCOM en deux grandes catégories : spatiales et terrestres. Les infrastructures spatiales regroupent les éléments nécessaires pour déployer, maintenir, suivre et contrôler un satellite. Quant aux infrastructures terrestres, elles concernent les éléments requis pour accéder à un répéteur satellite depuis les terminaux de station terrestre.

Ces terminaux de station terrestre englobent les équipements installés sur terre et à bord des avions et des navires et concernent donc aussi les territoires aériens et marins. C'est sur cette partie spécifique du segment terrestre qu'a porté la recherche de vulnérabilités.

IOActive a piloté la phase initiale d'un projet de recherche SATCOM interne. Cette phase s'est concentrée sur l'analyse et la rétro-ingénierie (reverse engineering) de mises à jour de firmware disponibles gratuitement et publiquement pour des technologies SATCOM développées et/ou commercialisées par des acteurs de premier ordre tels que Harris, Hughes, Cobham, Thuraya, JRC et Iridium.

L'objectif premier était de procéder à une évaluation de l'état de sécurité des terminaux SATCOM les plus largement déployés que sont Inmarsat et Iridium. Le chercheur a ainsi analysé les équipements utilisés pour accéder aux services suivants :

Inmarsat-C

Ce système de communication maritime fournit des services navire-terre, terre-navire et navire-navire. Ses fonctionnalités d'enregistrement et de transfert permettent d'utiliser le télex, le fax, la communication de données ou la messagerie électronique. Il s'agit d'un élément fondamental du Système mondial de détresse et de sécurité en mer (SMDSM), un ensemble de procédures, de types d'équipements et de protocoles de communication défini au niveau international pour renforcer la sécurité et garantir une réaction rapide et automatisée des autorités et des services de secours pour les situations de

détresse en mer. La convention internationale sur la Sauvegarde de la vie humaine en mer (SOLAS) rend obligatoire la présence d'équipements conformes SMDSM sur tous les navires marchands d'une capacité supérieure à 300 tonneaux de jauge brute (GRT).

VSAT

Les systèmes VSAT (terminal à très petite ouverture d'antenne) utilisent des transpondeurs satellites qui fonctionnent généralement dans les bandes C et Ku pour transmettre des données, de la vidéo ou la voix.

BGAN

Le réseau mondial de communication large bande (BGAN) est un réseau satellitaire d'envergure mondiale pour Internet et la voix. Grâce à des options de sécurité intégrées, ce service est adapté aux activités militaires.

BGAN M2M

Ce service mondial et bidirectionnel de données sur IP est conçu pour la gestion machine-machine (M2M) sur le long terme des actifs fixes. Il est couramment utilisé dans le secteur des systèmes de contrôle industriel (ICS) et pour les applications SCADA.

FB

FleetBroadband (FB) est un système satellitaire maritime large bande sur IP pour transmettre les données et la voix pour les communications opérationnelles et de l'équipage. Les systèmes de navigation modernes embarqués sur les navires, tels que le système de visualisation des cartes électroniques et d'information (ECDIS), peuvent s'appuyer sur la connexion de données fournie par ce service pour fonctionner correctement. Le système d'information de navigation informatisé ECDIS est conforme à la réglementation de l'Organisation maritime internationale (OMI) et peut s'utiliser en lieu et place des cartes maritimes papier.

SwiftBroadband

Il s'agit d'un système satellite aéronautique large bande sur IP pour la transmission des données et de la voix. Il est approuvé par l'organisation de l'aviation civile internationale (OACI) pour les services de sécurité aéronautique et joue un rôle important au sein des futurs systèmes de navigation aérienne (FANS).

Services Classic Aero Service

est un système de communication par satellite aéronautique pour la voix, le fax et les données. Il comprend les services suivants :

- Aero H, service multicanal voix, fax et données à 10,5 kbit/s fourni via une antenne à gain élevé dans les faisceaux globaux des satellites. Homologation OACI pour les services de sécurité.
- Aero H+, service multicanal voix, fax et données à 10,5 kbit/s fourni via une antenne à gain élevé dans les faisceaux étroits des satellites Inmarsat-3 et dans l'empreinte complète du satellite de la région de l'Océan atlantique (AOR) Inmarsat-4, pour un coût inférieur par connexion. Homologation OACI pour les services de sécurité.
- Aero I, service multicanal voix, données et fax en mode circuit à 4,8 kbit/s via une antenne à gain intermédiaire. Prise en charge de paquets de données à bas débit. Disponible dans les faisceaux étroits des satellites Inmarsat-3 et dans l'empreinte complète du satellite AOR Inmarsat-3. Homologation OACI pour les services de sécurité.
- Mini M Aero, service monocanal voix, fax ou données à 2,4 kbit/s pour l'aviation générale et les avions d'affaires de plus petite taille.

Ruben Santamarta précise le contexte de ses recherches : " *Il n'était pas envisageable de faire l'acquisition de chacun des équipements à analyser nous n'avions pas le budget, l'estimation était colossale. C'est donc sans accéder physiquement à l'équipement réel que nous avons fait une analyse statique du firmware, par rétro-ingénierie de tous les équipements. Nos travaux n'avaient pas pour objectif de contraindre le logiciel afin d'y détecter la corruption de mémoire classique, mais plutôt de comprendre les forces et les faiblesses de la sécurité native de ces équipements.*"

Par ce biais il a découvert des vulnérabilités reposant sur une multitude de backdoors (portes dérobées), de certificats codés en dur, de protocoles non documentés et non sécurisés ainsi que des algorithmes de chiffrement médiocres. Ces vulnérabilités permettent à des attaquants distants et non authentifiés de compromettre les produits concernés. Dans certains cas, aucune interaction de l'utilisateur n'est requise pour exploiter la vulnérabilité. L'envoi d'un simple SMS ou d'un message rédigé de manière spécifique depuis un navire à un autre suffirait à exploiter certains systèmes SATCOM.

En plus des failles de conception, IOActive a également découvert des fonctionnalités délibérément introduites dans les équipements. Ruben déclare : " *Je ne recommande à aucun concepteur d'introduire des backdoors, elles posent clairement de hauts risques pour la sécurité*". Dans son livre blanc l'éditeur détaille comment les attaquants pourraient tirer parti de ces vulnérabilités pour lancer différents type d'attaques, chaque scénario s'appuie sur la documentation du constructeur ainsi que sur l'usage et les déploiements réels des matériels.

Les travaux se sont concentrés sur deux domaines clés critiques : la collecte d'informations et la rétro-ingénierie (reverse engineering).

La meilleure façon de trouver le moyen de compromettre la sécurité d'un système est d'en comprendre le fonctionnement.

La collecte d'information est donc un élément fondamental. Au cours de cette phase, Ruben a collecté le maximum de données possible sur l'équipement concerné auprès de sources ouvertes et disponibles (fiches produits, guides de déploiement et de support, études de cas, manuels d'utilisation, commandes publiques, logiciels et firmware...)

Après avoir soigneusement analysé et évalué les données recueillies, Ruben était en mesure de savoir : comment le système a-t-il été conçu ? Quels sont ses composants ? Comment est-il habituellement déployé dans le cas de scénarios réels ? Quelles sont ses principales caractéristiques ?

À ce stade, il est en mesure d'estimer avec exactitude la surface d'attaque et de construire une carte des fonctionnalités. Disposer d'une liste détaillée de fonctionnalités est d'une aide inestimable pour la phase de rétro-ingénierie.

Nous avons affaire à un scénario où la cible principale, l'équipement physique, peut être inaccessible. Par conséquent, il nous faut compenser cet écueil en forçant l'analyse de deux composants fondamentaux : le logiciel de configuration et le firmware.

Les constructeurs fournissent généralement le logiciel « client » à leurs clients pour que ces derniers puissent configurer et contrôler le matériel. Par rétro-ingénierie de ces programmes, il est alors possible de comprendre comment l'équipement en question compte communiquer avec le monde extérieur et les types de protocole et/ou les mécanismes propriétaires impliqués dans ce processus.

IOActive développe souvent un équipement simulé pour feinter le logiciel de configuration et lui faire croire qu'il est réellement connecté à un véritable équipement. Nous optimisons cet environnement afin de collecter l'ensemble de données qu'un équipement est censé accepter ainsi que les résultats que le logiciel de configuration est censé attendre. Ces informations sont ensuite exploitées pour analyser le firmware.

Alors que le logiciel de configuration est généralement développé pour les trois principales plateformes que sont Windows, Mac ou Linux, le firmware est plus hétérogène. Il existe beaucoup de composants concernés, dont de nombreux systèmes d'exploitation en temps réel (RTOS), processeurs, cartes, puces, périphériques et interfaces.

Ceci rend fastidieuse l'analyse du firmware, laquelle exige des connaissances très spéci-

© Black Hat USA 2014.

fiques. L'approche de base implique de reconstruire les symboles, les références de chaînes de caractères et les cartes de la mémoire. Le firmware subit ensuite une opération de rétro-ingénierie afin de :

- Faire correspondre les fonctionnalités avec le code,
- Découvrir les fonctionnalités non documentées,
- Découvrir la communication entre les différents composants,
- Identifier les points d'entrée.

Dans certains cas, il est possible d'émuler des morceaux de code spécifiques présents dans le firmware. Ainsi, même sans accéder à l'équipement physique, nous pouvons savoir avec précision comment sont véritablement déployées certaines fonctionnalités. La solide compréhension des entrailles du firmware permet de rechercher les problèmes de sécurité et de mettre au point des vecteurs d'attaque.

Cette méthodologie d'analyse statique des équipements embarqués a été appliquée avec succès de nombreuses fois. En conséquence, l'éditeur a découvert de sérieuses vulnérabilités au sein d'équipements utilisés pour des systèmes de contrôle industriel, les communications par satellite ou des infrastructures de mesure de pointe.

Les menaces critiques que posent ces vulnérabilités techniques doivent éveiller les entités commerciales qui ne doivent pas minimiser la gravité des risques encourus par des entreprises tributaires de l'intégrité et de la confidentialité de ce type de communication. Il est évident que certains des services auxquels accèdent ces produits sont stratégiques au plan sécuritaire. Au vu des secteurs qui déploient ces produits, et les constructeurs concernés, la nature spécifique des vulnérabilités découvertes par IOActive est très inquiétante. L'état de sécurité actuel des produits analysés révèle qu'il est quasiment impossible de garantir l'intégrité de milliers d'équipements de communication par satellite. IOActive conseille la mise en oeuvre d'une action appropriée pour atténuer ces vulnérabilités. Selon elle, les détenteurs et fournisseurs de ces équipements devraient évaluer l'exposition sur le réseau de ces équipements, déployer des politiques fiables, segmenter le réseau et appliquer des modèles de flux du trafic (TFT) restrictifs dans les meilleurs délais.

A ce jour aucune réponse officielle n'a été apportée, ni même de projet de correctifs. L'objectif de ces recherches a pour but d'encourager les constructeurs à fournir des solutions de contournement officielles en plus des configurations recommandées afin de réduire au minimum le risque posé par ces vulnérabilités.

Ruben insiste : "*Si un seul des équipements concernés peut être compromis, c'est l'infrastructure SATCOM toute entière qui pourrait être en danger. Les navires, les avions, le personnel militaire, les services de secours, les médias, les sites industriels (plates-formes pétrolières, gazoducs, stations de traitement des eaux, éoliennes, sous-stations, etc.) pourraient tous être affectés par ces vulnérabilités.*"

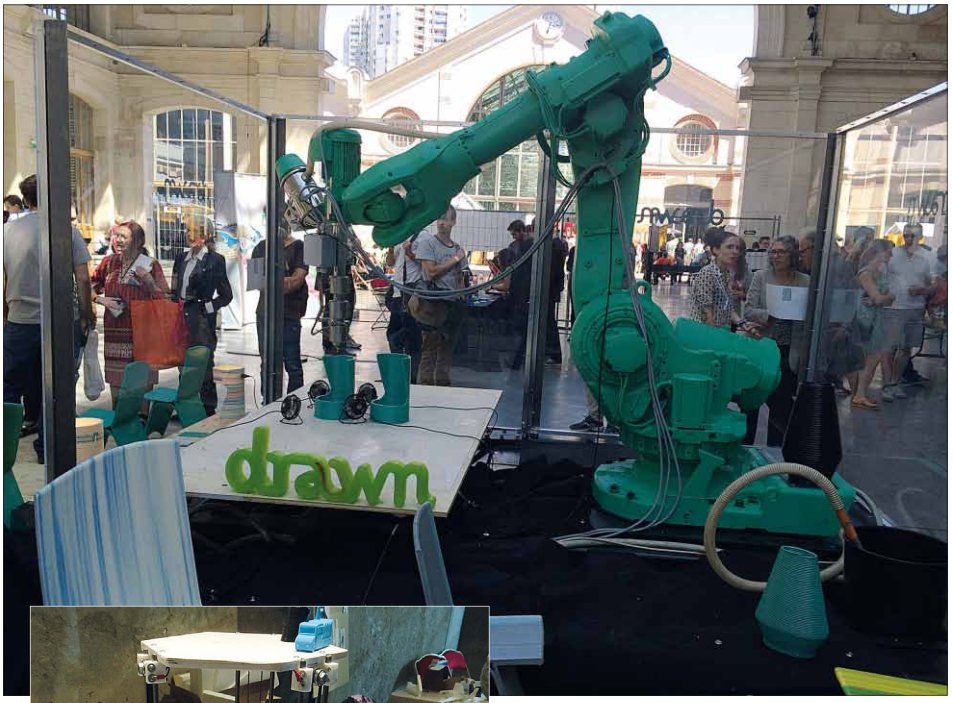
OActive est un promoteur de longue date de la divulgation responsable (responsible disclosure) des failles de sécurité. Jennifer Steffens, CEO de IOActive indique :"Depuis notre création, nous avons coordonné la communication de nos recherches et des conclusions critiques directement avec les fournisseurs, les CERTs et d'autres organismes de réglementation. Nous nous appuyons sur une large équipe internationale de chercheurs professionnels, nous travaillons avec les fournisseurs et les organismes de réglementation pour aider à l'éducation d'une part et d'autre part améliorer la sécurité des produits et technologies fréquemment utilisés par les consommateurs. Nous réussissons car nous exerçons nos activités et la gestion des vulnérabilités sensibles de manière éthique et responsable. Si, grâce à notre recherche indépendante, des vulnérabilités sont découvertes, IOActive travaillera à l'atténuation de la menace potentielle avec les entités concernées. Il est essentiel que toutes les parties concernées comprennent la nécessité du Responsible Disclosure et agissent en conséquence. Les fournisseurs doivent répondre rapidement et dans le meilleur intérêt de leurs utilisateurs. Nous devons tous nous rappeler que nous partageons l'objectif commun d'apporter une technologie sûre, pour tout le monde."

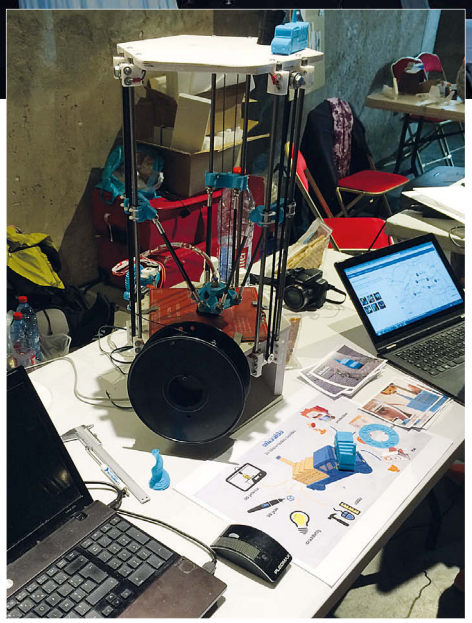
Présentation de Ruben Santamarta à Blac Hat "SATCOM Terminals Hacking by Air, Sea, and Land" : https://www.blackhat.com/docs/us-14/materials/us-14-Santamarta-SATCOM-Terminals-Hacking-By-Air-Sea-And-Land.pdf

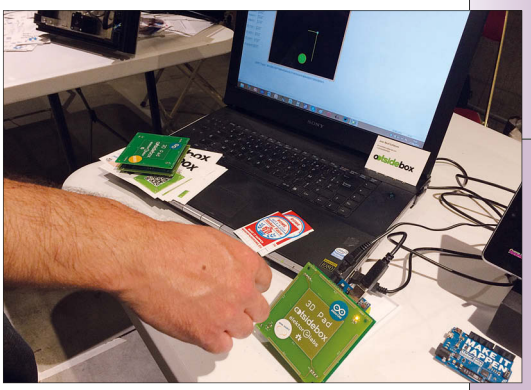
Véronique Loquet
Fondatrice de l'agence RP AL'X Communication. Spécialiste de l'Open Source et de la sécurité des SI depuis 1998, elle participe à de nombreuses conférences internationales et a co-fondé No Such Con Paris www.nosuchcon.org
sur Twitter @vloquet

Maker Faire Paris : notre sélection

En juin dernier, Paris accueillait le mouvement Maker Faire. Nous avons pu y voir de nombreux projets et constructeurs, grands et petits. Un foisonnement qui nous a beaucoup plu. Au lieu de faire le traditionnel compte-rendu, nous avons voulu retenir quelques exposants.







Des Arduino taillés pour l'éducation et un « Leap Motion » à monter !

La découverte de l'électronique était un des thèmes visibles durant cette Maker Faire Paris. Plusieurs stands utilisaient une déclinaison un peu spéciale d'Arduino : l'Educaduino. Cette plate-forme est issue d'un développement français mené par La Fonderie et Eurosmart. L'ambition est de rattraper notre retard sur l'électronique et la formation aux technologies du numérique auprès du public, des jeunes et à l'école. Il faut voir cette plate-forme comme un « mécano » moderne. Il se programme comme un Arduino. Il est vendu 30 €. Dans la même idée éducative, on pouvait aussi faire jouer les enfants au stand platypi et l'atelier « mon premier hack ». Le but : monter un petit circuit électronique et découvrir le microcontrôleur, un capteur, une LED. Nous avons adoré !

site : http://www.platypi.cc

Ootsidebox est un projet atypique : créer une plate-forme sans contact reconnaissant les gestes, un peu comme Leap Motion ou Kinect. Il utilise une base Arduino sur laquelle on rajoute le shield 3Dpad. Il est possible de l'utiliser au travers de tissus, du papier, un plastique. Le 3Dpad détecte les mouvements (à condition d'être à une distance « utile ». Un kit de développement est disponible pour intégrer ootsidebox dans son application. Un très beau projet, espérons qu'il se réalisera !

site : http://www.ootsidebox.com

Les objets connectés étaient bien entendu présents durant la Maker Faire. WeIO était un des représentants de cette forte tendance. WeIO est une plate-forme électronique et logicielle pour créer des objets connectés. La plate-forme dispose d'entrées/sorties, d'un SOC, d'un port micro-SD, du wifi. La fondation logicielle comprend un noyau Linux, SSH, Samba, de l'USB, d'un serveur Python. Pour développer, un IDE est disponible.

A découvrir : http://www.we-io.net

Impression 3D : vers la micro-usine personnelle ?

L'impression 3D révolutionnera notre quotidien et notre lien avec les objets. Elle va remettre en cause tout un pan de l'économie actuelle et de la notion d'obsolescence et pourrait même aider à réduire la société de consommation en favorisant les réparations personnelles et créer de véritables objets à la demande, chez soi ou dans des magasins équipés.

Durant cette Maker Faire, nous avons pu voir de nombreux constructeurs d'imprimantes 3D, notamment des modèles portatifs, à partir de 6-700 €. Oui, ces start-ups ne seront peut-être plus là dans 18 mois, mais le mouvement est engagé ! Une des tendances dans l'impression 3D est la création de modèles avec des buses d'impression fixées sur une tête très mobile capable de s'orienter selon différents axes. Un des meilleurs exemples était le modèle µdelta du constructeur eMotion Tech.

Dans la catégorie supérieure, nous trouvons l'imprimante grand format drawn. Son originalité est de pouvoir imprimer du mobilier sur mesure et de créer les objets les plus divers. Imaginez pouvoir imprimer votre table, votre bureau ! C'était une attraction du Maker Faire ! Site : http://www.drawn.fr

La robotique occupait une part non négligeable de l'événement. Nous y reviendrons dans un prochain numéro.

La rédaction.

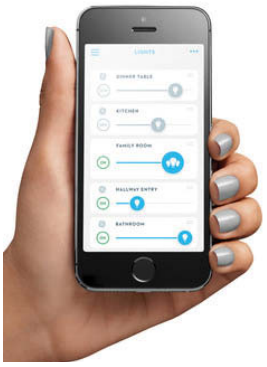
Intel investit dans la voiture connectée

La voiture connectée est décidément la nouvelle tendance pour les constructeurs et éditeurs informatiques. Intel a annoncé un projet de recherche commun avec Ford : le projet Mobile Interior Imaging (Mobii). Comme d'autres projets, Mobii a pour ambition de concevoir une nouvelle expérience utilisateur du conducteur. « Avec la recherche Mobii, notre objectif est d'explorer les diverses interactions des automobilistes avec les technologies présentes au sein du véhicule, afin de les rendre plus intuitives et plus prédictives, » déclare Paul Mascarenas, directeur de la technologie et vice-président de Ford Research and Innovation. « L'exploitation d'images captées à l'intérieur du véhicule n'en est qu'au stade de la recherche, mais les résultats que nous obtenons devraient nous permettre de modifier l'expérience consommateur sur le long terme. »
Mobii utilisera des caméras intérieures pour authentifier le conducteur et permettre de configurer automatiquement des fonctions de la voiture, les sièges, etc. La reconnaissance de geste pourrait permettre d'interagir avec la voiture.

Intel (suite) : un nouveau processeur Phi

Le processeur Phi est une nouvelle génération de processeurs créée pour les calculs massifs, les calculs massivement parallèles. La première génération avait souffert de plusieurs problèmes de performances et des difficultés à concevoir des applications adaptées.
Ce nouveau Xeon Phi doit améliorer les performances brutes avec l'intégration de la « fabric Intel Omni Scale ». Ces blocs assurent des transferts à très hautes vitesses.
Ce changement devrait se faire sans modification du code source actuel.
Disponibilité : 2e semestre 2015.

General Electric veut des ampoules connectées pas chères



Le problème avec les ampoules connectées, manipulables à distances est le prix de vente, parfois prohibitif. Le constructeur américain veut casser les prix avec ses ampoules Link (ampoules LED). Elles fournissent moins de fonctions que les ampoules HUE mais la Link démarre à 15 $/unité ! La gamme Link est disponible uniquement aux États-Unis. Link utilise le protocole ZigBee.
Pour les utiliser, il faudra passer par l'application Wink. Les fonctions seront basiques : allumer, éteindre... Link sera-t-il compatible HomeKit d'Apple ? Il faudra attendre l'automne pour le savoir...

UN MINI-PC POUR GAMER

Le Brix Gaming, de Gigabyte, est un mini-PC pour les joueurs. Il embarque un processeur i5 et une carte GeForce GTX 760 (architecture Kepler). Le boîtier peut recevoir jusqu'à 16 Go de mémoire et propose un connecteur mSata pour un SSD. Côté vidéo, il propose du DisplayPort, HDMI 1.4.
Gigabyte surfe sur les tendances barebone d'Intel NUC et surtout sur le concept de la Stream Machine. Il pourra faire tourner de nombreux jeux récents, les vidéos 4 K. Le constructeur n'a donné aucune date de sortie...

PowerColor Devil 13 R9 290X : un monstre à 1 400 $

Si vous aimez les monstres de puissances, la carte Devil 13 est sans doute pour vous, à condition de dépenser 1 400 $! Et c'est du lourd : 3 énormes ventilateurs, 3 emplacements occupés, + 2 kg, 2 processeurs Radeon R9 290X, 8 Go de mémoire vidéo. Bref, elle est énorme et nécessitera une solide alimentation pour ne pas faire flancher son PC. Il fournit une excellence puissance pour du 4 K et les jeux les plus récents, mais la carte dégage beaucoup de chaleur (les différents tests publiés par hexus et hardwarecanuks l'indiquent clairement). Malgré sa vélocité, faut-il réellement investir dans une telle carte ?

Une BBOX embarquant AndroidTV

Bouygues Telecom prépare une nouvelle BBOX basée sur le système AndroidTV. L'objectif est de proposer les fonctions de TV classiques et les fonctions propres à Android (applications, interface, accès web). Google et l'opérateur travaillent ensemble sur ce projet. La box embarquera un processeur 4 cœurs, 16 Go de stockage, 2 Go de mémoire vive, HDMI 1.4, USB, lecteur SD, support de Chromecast.
Disponibilité : fin 2014.

Timeline : 1984

Objet : MSX doit standardiser l'informatique grand public

A l'aide de ses connexions au Japon, Microsoft tentera de faire de l'ordinateur personnel un appareil grand public aussi universel que la chaîne hifi ou le magnétoscope. Mais les asiatiques se casseront les dents sur les marchés occidentaux.

Kazuhiko Nishi, Bill Gates et Paul Allen à la fin des années 70.

Le Spectravideo SVI-318 servira de modèle pour concevoir le standard MSX.

Le MSX V-20 de Canon.

Si tout s'était passé comme prévu, l'ordinateur personnel aurait dû devenir dans les années 80 un appareil électronique aussi standard qu'un magnétoscope ou qu'une chaîne hifi. Un truc qu'on branche sur la télé, pour jouer ou taper du texte à imprimer, avec des logiciels sur cartouche capables de fonctionner sur n'importe quel modèle de n'importe quelle marque, comme c'était déjà le cas dans l'industrie du disque et de la cassette vidéo. C'est en tout cas la vision qu'a Bill Gates en cette fin d'année 1977, lors d'un salon dédié à l'informatique, où il discute neuf heures durant avec un jeune passionné japonais, Kazuhiko Nishi, venu tout droit de Tokyo pour l'occasion. Également âgé de 21 ans, Kazuhiko Nishi a le même goût d'entreprendre ; en 1976, il avait voulu vendre un nouveau jeu vidéo, programmé avec des amis sur le matériel de la borne Pong. Puis, ne parvenant pas à acheter les composants, il s'était lancé dans la publication de magazines de vulgarisation sur la programmation. Bill Gates l'intéresse, car sa société a mis au point un Basic capable de fonctionner sur différents matériels. De son côté, Bill Gates est totalement excité par l'opportunité de faire affaire avec des japonais, ces gens qui n'ont par leur pareil pour faire entrer dans les foyers occidentaux des appareils électroniques standardisés, comme le magnétoscope VHS. En deux temps, trois mouvements, l'affaire est conclue : Kazuhiko Nishi, par l'intermédiaire de sa société d'édition ASCII, sera le représentant de Microsoft au Japon, et sa tâche sera de trouver les partenaires industriels locaux pour plancher sur un ordinateur domestique standardisé. Kazuhiko Nishi ne va en fait rien trouver pendant 5 ans, mais ASCII servira tout de même à revendre le Basic de Microsoft à quelques industriels japonais pressés de reproduire chez eux le phénomène micro-ordinateur. C'est ainsi que naissent les PC 8001 et 8801 (alias PC-88) de Nec. Mais aucun standard n'en découle.

Le MSX CX5M de Yamaha est le premier micro-ordinateur capable de piloter un clavier midi.

SVI-318, l'ancêtre

Puis, en 1982, le ministère de la Poste japonais annonce avoir commandé au groupe Matsushita (marques Panasonic et JVC) des micro-ordinateurs standardisés, dans le but de créer un réseau informatique à l'image du réseau téléphonique. C'est l'occasion ou jamais. Kazuhiko Nishi part présenter à Matsushita le prototype du prochain micro-ordinateur américain qui doit intégrer le Basic de Microsoft en ROM, le SVI-318 de Spectravideo. Ce SVI-318 présente plusieurs intérêts. D'abord, il est moins cher et plus performant que le PC qu'IBM vient de lancer en Amérique. Il dispose ainsi d'un processeur Z80 à 3,58 MHz, d'une puce audio Yamaha AY-3-8910 qui laisse entendre des mélodies sur 3 voies et 7 octaves et, surtout, d'un chipset vidéo TI9928A qui affiche 256x192 pixels en 16 couleurs avec 32 sprites et, ce, dans 16 Ko de RAM dédiée. Ensuite, l'architecture du SVI-318 est déjà reprise par un autre constructeur, Coleco, pour concevoir sa console de salon, la Colecovision. Kazuhiko Nishi connaît bien le SVI-318 : la machine étant assemblée à Hong Kong, à une poignée d'heures de vol de chez lui, il s'est mêlé de conseiller Spectravideo sur sa conception. Matsushita est enchanté par le projet ! Le groupe fédère autour de lui tous les industriels asiatiques du secteur de l'électronique grand public (Canon, Casio, Fujitsu, Goldstar, Hitachi, Mitsubishi, Nec, Pioneer, Sanyo, Samsung, Sharp, Sony, Toshiba, Yamaha...), ainsi que son partenaire européen, Philips.

Et MS-DOS fut porté sur micros 8 bits

Aux USA, la nouvelle de ce partenariat est assez inattendue. Chez Spectravideo, le projet paraît si colossal qu'on préfère laisser Microsoft (ASCII en l'occurrence) se charger de revendre lui-même des licences du concept à tout ce beau monde. Du coup, le projet est nommé MSX, pour Micro-Soft eXtended ; l'éditeur ayant promis, par la voix de son représentant japonais, que ce qu'il livrait-là serait supérieur (« étendu ») à ce que proposait IBM dans son PC, fonctionnant sous le MS-DOS de Microsoft. Cela dit, les interprétations divergent. D'autres prétendent que MSX signifierait tout autant Machines with Software eXchangeability.Chez Microsoft, on est aussi pris de court. Personne n'avait prévu de MS-DOS pour le MSX. Alors, au moment de l'annonce des machines, durant l'été 1983, Paul Allen, le cofondateur de l'éditeur, retourne voir Tim Paterson, le créateur

de MS-DOS, pour qu'il adapte vite-fait une version de MS-DOS au processeur Z80 des MSX. MS-DOS étant déjà une réécriture plus ou moins pirate d'un système conçu pour Z80, le fameux CP/M de Digital Research, le portage est trivial et prend à peine deux mois (mais il est facturé 100.000 dollars à Microsoft). Le résultat est un MS-DOS 1.25 pour MSX, rapidement renommé MSX-DOS. Dans un premier temps, MSX-DOS doit prendre la forme d'une ROM de 32 Ko, insérée dans le port cartouche et reliée à un lecteur de disquettes. Lorsqu'elle est installée, cette ROM prend le contrôle du MSX au démarrage et cherche à booter une disquette ; si elle n'y parvient pas, elle propose de basculer au Basic classique ou vers une ligne de commande MS-DOS standard.

La laborieuse conquête du marché occidental

Les premiers micros MSX sortent sans disquette entre fin 1983 et mi-1984, principalement à cause d'ASCII qui se mêle de retravailler MSX-DOS. Du coup, les logiciels sont livrés sur cartouche. Les marques se différencient par des design ou des extensions propres : le Yamaha CX5M peut par exemple piloter un clavier Midi, tandis que le Pioneer Palcom PX-7 permet d'afficher du texte ou des graphiques par-dessus l'image d'un vidéodisque. Aux USA, le MSX fait un bide. Aucun constructeur historique ne veut adhérer au standard - à part Spectravideo, mais un an plus tard, lorsqu'il aura écoulé ses SVI-318 qui ne sont pas compatibles - et Commodore tire les prix vers le bas au point de décourager tous les constructeurs japonais d'exporter leur machine. Un MSX vaut alors en moyenne 2,5 fois le prix d'un C64. En Europe, sous l'impulsion de Philips, le MSX est plutôt bien accueilli par la presse, mais les éditeurs locaux commettent l'erreur d'y adapter leurs titres ZX Spectrum, machine britannique la plus proche techniquement (même processeur). Hélas, la mémoire vidéo dédiée du MSX, qui aurait dû être un avantage, s'accommode mal des jeux conçus pour la mémoire vidéo du ZX qui, elle, est partagée. Par conséquent, les adaptations tournent plus lentement sur MSX que leurs versions originales. Autre problème, comme personne n'a pensé à définir un adressage mémoire standard, plusieurs cartouches de jeux plantent aléatoirement selon les modèles de MSX. Boudé par les américains, jugé peu fiable par les européens, le MSX s'exporte ailleurs. En Union soviétique et dans les pays du Moyen-Orient, il est la seule machine adaptée aux alphabets locaux. Et pour cause : la gestion de différents caractères était une problématique de départ pour cette machine japonaise d'inspiration américaine. Il y sert autant d'ordinateur pour l'éducation que de machine à faire les sous-titres des cassettes vidéo pirates ; la puce vidéo TI9928A est nativement capable de mixer ses graphismes avec une source externe et il existe même une instruction en Basic (Call Impose) pour le faire. On trouve également des MSX en Amérique du Sud et, par extension, en Espagne, où l'on profite de la riche ludothèque hispanophone développée outre-Atlantique.

Le NMS8245, les MSX2 de Philips.

Le HitBit F500 de Sony, des MSX2 avec unité centrale séparée.

Le seul modèle de MSX Turbo-R sera fabriqué par Panasonic.

L'entêtement à produire un super 8 bits à l'époque des 16 bits

En 1985, Bill Gates voit son MS-DOS prospérer sur le marché occidental du PC grâce à l'envolée des cloneurs d'IBM. Comparativement, il juge que le MSX est un ratage commercial. Kazuhiko Nishi veut se rattraper. Il propose aux industriels japonais une nouvelle version : le MSX2, avec une puce graphique, le V9938 de Yamaha, qui permet d'atteindre 512x212 pixels en 16 couleurs, ou 256x212 pixels en 256 couleurs. Problème, le reste de la machine n'évolue pas, en particulier le processeur Z80 qui fait désormais bien pâle figure face aux tous derniers 16/32 bits des Atari ST et Commodore Amiga. Bill Gates, en a assez. Il propose de racheter ASCII pour recadrer la stratégie. Kazuhiko Nishi refuse. Bill Gates claque la porte du projet. De 23 fabricants au départ, il n'en reste plus que 14 en 1986 pour vendre le MSX2. Les problèmes de lenteur graphique sont résolus, le lecteur de disquettes 3,5 pouces se généralise, la RAM grimpe à 512 Ko et la RAM vidéo à 128 Ko. Hélas, comme l'avait prévu Bill Gates, la machine paraît déjà ringarde. C'est un re-bide. Rebelote en 1988 : il n'y a plus que 5 constructeurs pour vendre le MSX2+ qui affiche désormais 19268 couleurs, mais traîne toujours son vieux Z80 à 3,58 MHz. Re-re-bide. Kazuhiko Nishi ne jette pas l'éponge : il engage des ingénieurs pour développer au sein d'ASCII-même un processeur RISC R800 compatible avec le Z80, mais quatre fois plus rapide à 7,16 MHz. En vain : le MSX3 ne sortira jamais, faute de nouveau processeur graphique. Panasonic lancera tout de même en 1991, et uniquement au Japon, un MSX Turbo-R doté du R800 et de la puce graphique du MSX2+. Puis jettera aussi l'éponge. ASCII vivotera jusqu'en 1996 des ventes de ses magazines et de ses jeux pour Nintendo et Sega, puis sera fusionné avec d'autres entreprises. De son côté, Bill Gates mènera à bout son projet de micro-ordinateur universel, avec le lancement de Windows 95. Sur PC.

Yann **Serra**

Abonnement : Programmez, 17, Route des Boulangers, 78926 Yvelines Cedex 9 - 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 : 30 € (Monde Entier) souscription exclusivement sur www.programmez.com

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

Ont collaboré à ce numéro : Sylvain Saurel, F. Bordage, J. Chatard, Y. Serra

Secrétaire de rédaction : Olivier Pavie

Experts : M. Sylla, M. Garcia, M. Chaize, O. Guilloux, M. Rousseau, D. Catuhe, P. Fernandez Duran, A. Morgaut, A. Lemar-Verrier, K. Sibué, S. Sibué, F. Fadel, G. Nieutin, S. Ollivier, P-A Gury, J. Corioland, S. Beaupuis, A. Pacaud, V. Kovalsky David, C. Villeneuve, C. Chervy, N. Grekas, A. 'Peck' Solleiro, V. Loquet.

Crédits couverture : © SensorSpot / istock
© fotografiedk / Fotolia.com

Maquette : Pierre Sandré

Publicité : PC Presse,
Tél.: 01 74 70 16 30, Fax : 01 41 38 29 75
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 : 0967320934

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 : 1215 K 78366 - ISSN : 1627-0908

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

WINDEV 19
DÉVELOPPEZ 10 FOIS PLUS VITE

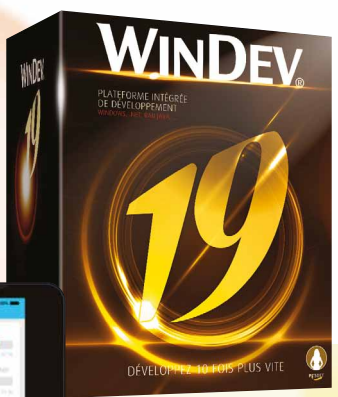
WINDEV : Logiciel professionnel
de développement multi-plateformes

Vous méritez le meilleur, vos équipes méritent le meilleur pour être le plus efficace possible, quel que soit le matériel sur lequel vos applications vont fonctionner : **PC**, **tablette**, **smartphone,** en application native ou en **site Internet**.

Incroyable : grâce à **WINDEV 19**, **WEBDEV 19** et **WINDEV Mobile 19**, **les applications que vous écrivez sont nativement portables**.

Vous réutilisez votre code, vos fenêtres, vos états... dans tous les environnements: **Windows**, Mac, Linux, **Android**, iOS (**iPhone**, **iPad**), **Windows Phone**, pour des applications natives et pour des sites, avec les données en local, sur serveur local ou distant, ou encore dans le **cloud**.

Dans l'intérêt de votre entreprise, dans l'intérêt de vos utilisateurs et de vos clients, commandez aujourd'hui votre **WINDEV 19** !



Environnement de développement professionnel depuis 20 ans

Dossier + DVD + Témoignages sur simple demande (gratuit)

Fournisseur Officiel de la Préparation Olympique

WINDEV AGL N°1 en FRANCE

www.pcsoft.fr

Des centaines de références sur le site