Developpez

Le Mag

Édition de décembre - janvier 2013/2014.
Numéro 49.
Magazine en ligne gratuit.
Diffusion de copies conformes à l'original autorisée.
Réalisation : Alexandre Pottiez
Rédaction : la rédaction de Developpez
Contact : magazine@redaction-developpez.com

Sommaire

Éditorial

Avec la nouvelle année, vient toujours un nouveau numéro du magazine du club des développeurs.

La rédaction vous présente ses meilleurs vœux et plein de merveilleux projets pour cette nouvelle année !

La rédaction

Article Perl

Programmation fonctionnelle en Perl – Partie 1 : Les opérateurs de listes

Ce document est le premier d'une série de tutoriels visant à montrer comment utiliser certaines techniques de la programmation fonctionnelle en Perl.

par **Laurent Rosenfeld**
Page 9

Article Réseau

L'internet Rapide et Permanent – La technologie Wi-Fi

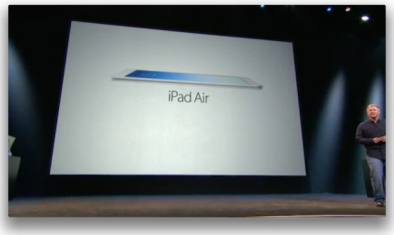
Cet article parlera de la technologie liée aux réseaux WiFi.

par **Christian Caleca**
Page 34

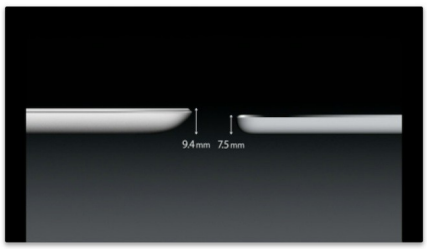
Keynote Apple 22/10 : Nouveaux iPad Air et iPad Mini Retina

Après avoir mis à jour les MacBook Pro, Mac Pro, OS X Mavericks et les suites iLifes et iWork, Apple dévoile ses nouveaux iPad.
L'iPad devient iPad Air et l'iPad Mini Retina vient gonfler le catalogue d'Apple.

iPad Air



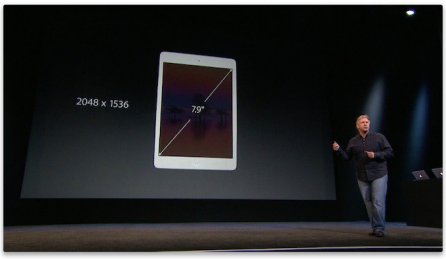
L'iPad Air est le nouvel iPad, toujours avec un écran 9,7 pouces mais une rupture avec le modèle précédent au niveau du poids (460 grammes contre 600 grammes pour l'iPad 4). Dotée d'un design proche de l'iPad Mini, la tablette est équipée de la puce A7 64 bits, est 20 % plus fine que la génération précédente et dispose d'une caméra 5 mégapixels, ainsi que d'une autonomie de 10 heures. On notera cependant l'absence de TouchID comme sur les iPhone 5S.

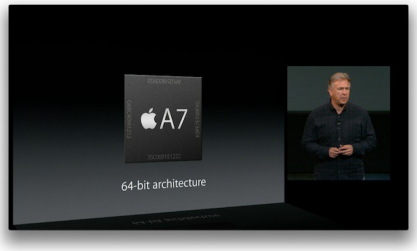


L'iPad Air sera disponible le 1er novembre dans tous les pays pour un prix de 499$ (environ 489 euros en France).

iPad Mini Retina

L'iPad Mini reçoit son écran Retina de 2048x1568 pixels, sa puce A7 64 bits est quatre fois plus rapide. La mini tablette possède 10 heures d'autonomie.





L'iPad Mini Retina est proposé à partir de 399$ pour la version 16 Go et l'ancien iPad Mini à 299$ (modèle 16 Go).

Avec ces nouveaux modèles de tablettes, Apple veut rester maitre sur un marché qui commence à lui filer entre les doigts. La société fait face à une concurrence importante des tablettes Android. L'annonce d'Apple coïncide avec la présentation de la première tablette de Nokia (Lien 01), qui se positionne comme un outsider à ne pas négliger.

Présentation de l'iPad Air : Lien 02

Présentation de l'iPad Mini : Lien 03

Commentez la news d'Aurélien Gaymay en ligne : Lien 04

Keynote Apple 22/10 : OS X Mavericks, iLife et iWork disponibles gratuitement

Lors de la Keynote d'Apple du 22 octobre, la société a présenté ses nouveaux logiciels pour cette fin d'année 2013 avec au programme : Mavericks, iLife et iWork.

OS X Mavericks

La nouvelle version de l'OS de la firme, Mavericks (10.9) avait déjà été présentée lors de la WWDC en juin et Apple l'avait promis pour tous pour le mois d'octobre. C'est aujourd'hui chose faite.

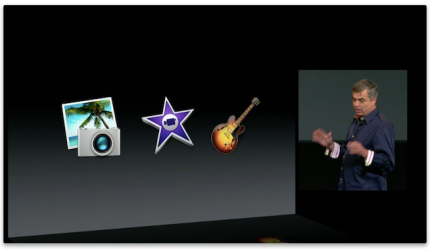
Apple a refait le tour des nouveautés, comme Plans, iBooks, Safari, les onglets dans le Finder ainsi que les Tags.
OS X Mavericks permet une amélioration de l'autonomie allant jusqu'à une heure d'autonomie en plus.
La mémoire est compressée et permet d'entrer 6 Go de données dans 4 Go de mémoire vive.
L'ajout de l'OpenCL pour les cartes graphiques intégrées est également au rendez-vous.

Craig Federighi montre le prix de vente de Windows 8 Pro qui est de 199$ (petit croc-en-jambe à Microsoft) et annonce que OS X Mavericks sera **disponible dès aujourd'hui** et **gratuitement** … Oui, GRATUIT !!

Pour les utilisateurs de Snow Leopard (10.6) la mise à jour sera aussi **gratuite**.

iLife

Selon Eddy Cue, c'est la plus grande mise à jour d'OS X : iPhoto, iMovie et GarageBand ont été revues, réécrites pour le 64 bits, ont reçu un lifting du design et sont désormais plus rapides.



La suite iLife est disponible dès maintenant et gratuite pour tous les nouveaux Mac et appareil iOS sur le Mac App Store et iTunes Store.

iWork

Toute la suite iWork a été réécrite en 64 bits et elle est donc plus rapide et est enfin disponible sur Mac après des années d'attente.

Pages, Numbers et Keynote ont toutes eu droit à une mise à jour graphique avec, par exemple, Pages qui dispose d'une barre latérale au lieu d'une barre flottante, un peu comme les dernières versions de Xcode.

Comme iLife, iWork est **disponible dès maintenant** et

aussi **gratuitement** pour tous les nouveaux Mac et appareils iOS sur le Mac AppStore et iTunes Store.

Présentation de OS X Mavericks : Lien 05

Commentez la news d'Aurélien Gaymay en ligne : Lien 06

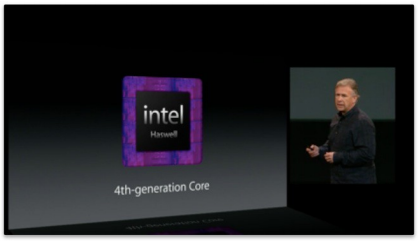
Keynote Apple 22/10 : Nouveau MacBook Pro et nouveau Mac Pro

Apple a présenté lors de sa dernière keynote, les nouveaux modèles de MacBook Pro 13" et 15" ainsi que les nouveaux Mac Pro. La gamme MacBook disparait pour ne laisser que les MacBook Air et MacBook Pro.

MacBook Pro

Phil Schiller, vice-président d'Apple, a présenté les nouveaux MacBook Pro 13 et 15 pouces, tout en vantant la légèreté et la diminution d'épaisseur (1,8 cm) de ceux-ci.

Reposant sur les nouveaux processeurs Haswell pour le modèle de 13 pouces et sur la nouvelle génération Intel, Crystalwell pour le second modèle, les nouveaux appareils d'Apple ont de quoi séduire les consommateurs.



Puces graphiques Intel Iris et Iris Pro jusqu'à 90 % plus rapides, SSD, Wifi (ac), Thunderbolt 2, sont quelques-unes des spécificités de ces dispositifs.

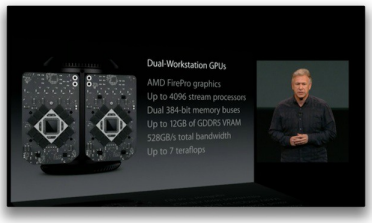
Les MacBook Pro 13 et 15 pouces voient leurs prix baissés de 200$ et ils sont disponibles dès maintenant pour respectivement 1299 dollars et 1999 dollars.

Lien 07

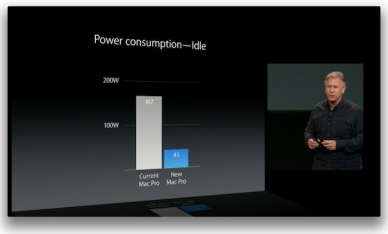
Mac Pro

La machine dédiée aux professionnels est maintenant plus petite et bien sûr plus puissante.

Elle offre jusqu'à 64 Go de RAM, deux puces graphiques AMD FirePro et jusqu'à 12 Go de mémoire dédiée.



Uniquement équipé de SSD (pas de disque dur), le Mac Pro dispose de 6 ports Thunderbolt 2, 4 ports USB 3, le support pour 3 écrans 4K et une consommation d'énergie de moins de 70 %. Il est plus silencieux que l'ancienne version.



Le Mac Pro sera disponible au mois de décembre pour le prix de 2999 dollars et est assemblé aux É.-U..

Présentation des nouveaux Mac : Lien 08

Commentez la news d'Aurélien Gaymay en ligne : Lien 09

Comment développer une application iOS multilingue

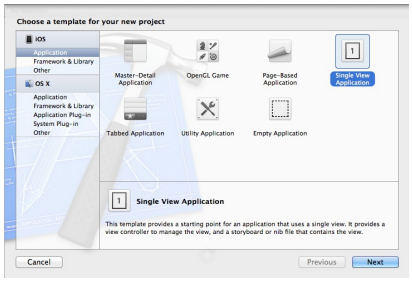
Trouver une application iOS multilingue sous Apple Store est de plus en plus fréquent. Dans ce tutoriel nous allons voir ensemble comment développer une application de ce genre.

1. Introduction

Trouver une application iOS multilingue sous Apple Store est de plus en plus fréquent. Dans ce tutoriel nous allons voir ensemble comment développer une application de ce genre.
Nous allons étudier l'utilisation avancée de NSLocalizedString. Elle consiste à afficher le contenu approprié selon le choix de la langue de l'utilisateur.

2. Création du projet sous xcode

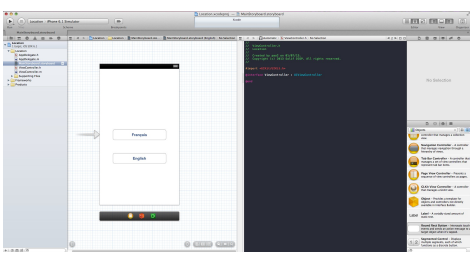
La première étape consiste à créer un projet xcode, pour cela lancez-le et créez un nouveau projet.



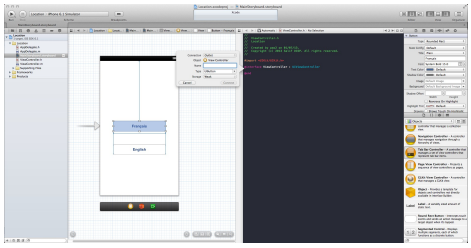
Faites **Next** puis donnez un nom à votre projet, dans mon exemple j'ai choisi de l'appeler **Location**.
Nous allons procéder maintenant à la mise en forme (très basique) c'est-à-dire : ajouter les boutons dont nous avons besoin et un peu de couleur. Rendez-vous maintenant sur la fenêtre de gauche qui liste tous vos fichiers. Cliquez sur le fichier **MainStoryboard.storyboard,** vous devriez obtenir la fenêtre ci-dessous :
Xcode vous a créé de base un projet contenant un ViewController. Ajoutez à cette vue deux **Round Rect Button**, puis nommez l'un français et l'autre english. En effet, l'application que nous sommes en train de créer permettra d'afficher le contenu approprié en français ou en anglais.

2.1. Ajout des méthodes liées à l'appui sur les boutons

Tout d'abord, cliquez sur le menu Editor situé en haut à droite de votre IDE qui vous permettra d'afficher le ficher d'entête lié à votre vue.



Pour ajouter une méthode à votre bouton : maintenez la touche **ctrl**, cliquez sur un bouton puis déplacez votre curseur vers le fichier d'entête ouvert sur votre droite. Placez le curseur de la souris entre @interface ViewController : UIViewController et @end, puis relâchez la souris et la touche **ctrl.** Une petite fenêtre semblable à celle-ci devrait apparaître :



2.2. Description de la fenêtre

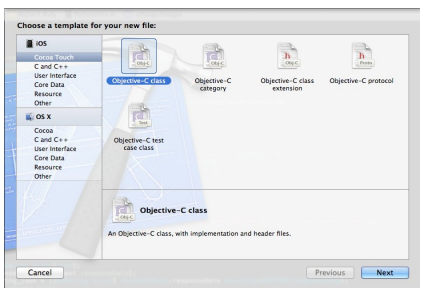
Connection : vous permet de choisir le type de connexion à votre bouton.
Object : spécifie le type d'objet.
Name : contient le nom de votre connexion. À vous de le choisir.
Type : c'est le type de l'objet sur lequel la connexion doit s'effectuer. Dans notre cas c'est un UIButton.
Storage : définit la manière dont notre objet est géré en mémoire.

2.3. Configuration de l'objet

Dans le menu déroulant Connection choisissez une connexion de type Action, puis donnez-lui un nom dans Name et terminez par OK. Une méthode est créée dans votre fichier d'entête.
Répétez cette procédure pour l'autre bouton.

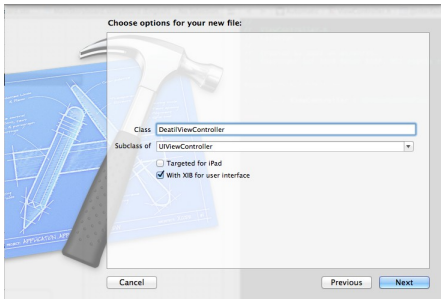
3. Création d'une classe incluant un fichant xib

Pour résumer, un fichier xib vous permet d'éditer graphiquement votre View comme on le fait avec le storyboard.
Allez sur le menu fichier -> new-> File… vous arrivez sur cette fenêtre :



Choisissez **Objective-C class**, faites **Next**.
Donnez un nom à votre classe dans le champ **class,** cochez

la case With XIB for user interface et pour finir dans le champ Subclass of choisissez UIViewController.



Faites Next pour finir et enregistrer votre classe.
Vous remarquerez qu'en plus de vos fichiers DetailViewController.m et DetailViewController.h vous avez un troisième fichier DetailViewController.xib, il a le même rôle que le storyboard. Il vous permet d'éditer graphiquement votre application.
Cliquez sur ce fichier une fois pour le sélectionner et ajoutez-y un objet UILabel, élargissez de façon à ce qu'il occupe la moitié de votre vue puis allez sur votre fenêtre située à droite de votre IDE afin d'accéder aux propriétés de votre UILabel, ajoutez un nombre de lignes supérieur à 1, pour ma part j'ai choisi cinq lignes. Répétez la même procédure qu'on a faite pour les deux boutons, mais en choisissant comme type de connexion **Outlet.** Donnez-lui un nom. Dans notre exemple, j'ai choisi de l'appeler quoteLabel.
Ouvrez ensuite le fichier .m correspondant et rajoutez entre @implementation DetailViewController :

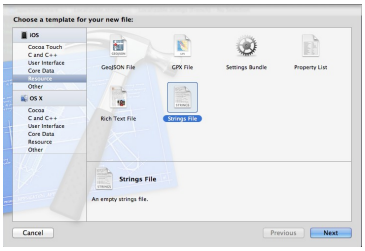
```
- (id)initWithNibName:(NSString *)nibNameOrNil
bundle:(NSBundle *)nibBundleOrNil

@synthesize quoteLabel;
```

3.1. Ajout des fichiers Localizable.Strings

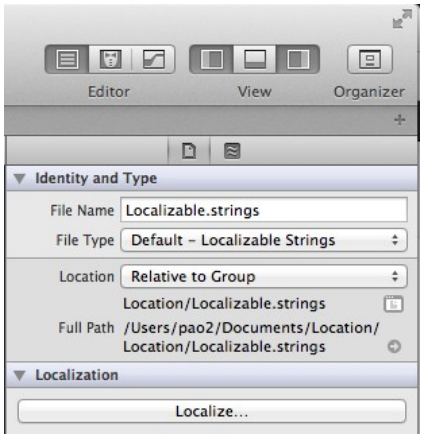
Les fichiers localizable.strings sont les fichiers de langue et vont contenir la traduction de notre application dans les différentes langues que l'on souhaite. Pour cet exemple nous allons nous limiter à deux fichiers Localizable.strings pour l'anglais et le français.

3.2. Quel est son fonctionnement ?

Les fichiers Localizable.strings contiennent une clé et sa valeur. La clé choisie doit être toujours la même dans tous les autres fichiers Localizable.strings. Elle doit respecter la syntaxe suivante : « [clé]» = « [valeur] » ;
Exemple :
"parcsjardins" = "Gardens and Parks";
Maintenant nous allons ajouter ces fichiers dans notre projet, pour cela, allez dans le menu **Fichier** (de xcode) **New -> File…**



Choisissez ensuite le menu Ressource d'iOS puis le type Strings File.
Faites ensuite **Next** pour enregistrer votre fichier et nommez-le **Localizable.strings.**
Vous avez maintenant votre fichier **Localizable.strings** qui apparaît dans la fenêtre, listant tous les fichiers de votre projet, à gauche de votre IDE. Nous allons maintenant l'affecter à une langue. Pour cela, cliquez sur le fichier et sur la partie gauche de vote IDE :



Comme le montre la capture d'écran ci-dessus, cliquez sur Localize… et choisissez la langue anglaise. Nous allons maintenant ajouter le fichier de **Localizable.string** pour la langue française.
Cliquez sur le nom de votre projet.
Allez dans l'onglet info, puis la section **Localizations** cliquez sur le plus et choisissez la langue française. Décochez les fichiers ressources MainStoryboard.storyboard et InfoPlist.strings pour ne garder que **Localizable.strings**. Validez votre opération et ça y est vous avez vos deux fichiers de localisation qui apparaissent sur la fenêtre gauche de votre projet.
Ouvrez maintenant le fichier Localizable.strings(english) pour y coller le texte ci-dessous :
"citation" = "« It's hard enough to find an error in your code when you're looking for it. It's even harder when you've assumed your code is error-free. » - Steve McConnell";
Puis dans le fichier Localizable.strings(french) celui-ci :
"citation" = "« Il est assez difficile de trouver une erreur dans son code quand on la cherche. C'est encore bien plus dur quand on est convaincu que le code est juste. » - Steve McConnell ";
Nous avons maintenant fini avec ces deux fichiers. Nous allons créer la classe qui va nous retourner le contenu adéquat selon la langue choisie.

4. Création de la classe LocalUser

Créer une nouvelle classe objective-c, faites-la hériter de NSObject en la choisissant dans le menu déroulant sub of class. Nommez votre classe, pour l'exemple j'ai choisi de l'appeler LocalUser. Cette classe va contenir les méthodes qui vont nous permettre de parcourir les fichiers Localizable.strings et de nous retourner le bon contenu selon la langue choisie.
Ouvrez le fichier .h de cette classe et déclarez les méthodes suivantes :

```
@interface LocalUser : NSObject
```

```
{
    NSBundle *bundle;
}

- (void)initialize;
- (void)setLanguage:(NSString *)langue;
- (NSString *)get:(NSString *)key alter:(NSString
*)alternate;
```

Puis dans le fichier .m leurs définitions :

```
#import "LocalUser.h"

@implementation LocalUser

/*
 * Méthode d'initialisation
 *
 */
- (void)initialize
{
    NSUserDefaults* defs = [NSUserDefaults
standardUserDefaults];
    NSArray* languages = [defs
objectForKey:@"AppleLanguages"];
    NSString *current = [languages
objectAtIndex:0];
    [self setLanguage:current];
}

/* Cette méthode permet de
 * définir le bon fichier de localizable.strings
 *
 */
- (void)setLanguage:(NSString *)langue
{

    NSString *path = [[ NSBundle mainBundle ]
pathForResource:langue ofType:@"lproj" ];
    bundle = [NSBundle bundleWithPath:path];
}

/* Cette méthode permet de récupérer la bonne
traduction
 * en fonction de la clé passée en paramètre*/
- (NSString *)get:(NSString *)key alter:(NSString
*)alternate
{
    NSString *word = [bundle
localizedStringForKey:key value:alternate
table:nil];
    return word;
}
@end
```

5. Édition de la classe ViewController

La classe ViewController est la classe qui a été générée lorsque vous avez créé votre projet. Nous allons ajouter dans le fichier .h la méthode

```
- (void) setLanguageAndRun:(NSString *)code;
```

Nous verrons à quoi elle nous sert dans le ficher .m.
Ouvrez maintenant le fichier ViewController.m, puis ajoutez le contenu ci-dessous :

```
#import "ViewController.h"
#import "DetailViewController.h"
#import "LocalUser.h"

// Nous importons les fichiers
DetailViewController.h et LocalUser.h afin
// de pouvoir les utiliser et faire appel à leurs
méthodes


@interface ViewController ()

@end

@implementation ViewController

- (void)viewDidLoad
{
    [super viewDidLoad];
    // Do any additional setup after loading the
view, typically from a nib.
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be
recreated.
}

- (IBAction)appuiFr:(id)sender {

    [self setLanguageAndRun:@"fr"];

}

- (IBAction)appuiEn:(id)sender {

     [self setLanguageAndRun:@"en"];
}

- (void)setLanguageAndRun:(NSString *)code
{
    LocalUser *myLocal = [[LocalUser alloc]init];
// on crée une instance de type de la classe
LocaUser puis on l'initialise

    [[NSUserDefaults standardUserDefaults]
setObject: [NSArray arrayWithObjects:code, nil]
forKey:@"AppleLanguages"]; // On met à jour la
langue choisie
    // par l'utilisateur
    [[NSUserDefaults standardUserDefaults]
synchronize];
    [myLocal setLanguage:code]; // on sauvegarde
ce choix
    DetailViewController *detailView =
[[DetailViewController
alloc]initWithNibName:@"DetailViewController"
bundle:nil]; // On crée une instance de la classe
DetailViewController qui
    //représente la vue DetailViewController
    [self.navigationController
pushViewController:detailView animated:YES]; //
On lance l'exécution de notre vue
}

@end
```

Explication :
Les méthodes - (IBAction)appuiFr:(id)sender et - (IBAction)appuiEn:(id)sender sont celles de nos deux boutons pour les langues anglaise et française. Elles

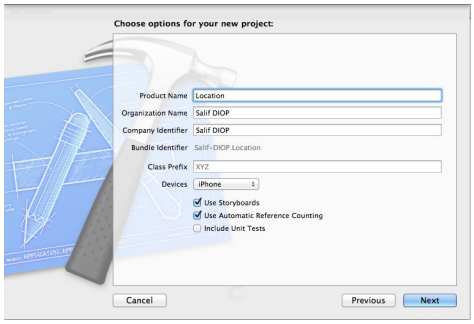
exécutent une ou plusieurs actions définies par vous-même lorsque l'utilisateur agit en appuyant sur l'un des boutons. Dans notre cas l'exécution de ces deux méthodes fait appel à - (void) setLanguageAndRun:(NSString *)code qui prend en paramètre le code de la langue choisie par l'utilisateur en pour la l'anglais et fr pour le français.
Attention le code représentant chaque langue est normalisé consulter ce lien pour plus de détails : Lien 10.

6. Édition de la classe DetailViewController

C'est presque fini, c'est la dernière classe à éditer et nous aurons terminé notre projet.
Ouvrez le fichier DetailViewController.m afin d'y ajouter le code ci-dessous :

```
#import "DetailViewController.h"
#import "LocalUser.h"

@interface DetailViewController ()

@end

@implementation DetailViewController
@synthesize quoteLabel; // le nom de notre label qui nous sert pour l'affichage

- (id)initWithNibName:(NSString *)nibNameOrNil bundle:(NSBundle *)nibBundleOrNil
{
    self = [super initWithNibName:nibNameOrNil bundle:nibBundleOrNil];
    if (self) {
        // Custom initialization
    }
    return self;
}

- (void)viewDidLoad
{
    [super viewDidLoad];

    LocalUser *myLocal = [[LocalUser alloc]init]; // on instancie un nouveau de type de la classe LocalUser

    [myLocal initialize]; // On fait appel à la méthode initialize

    quoteLabel.text = [myLocal get:@"citation" alter:@""]; // on fait appel à la méthode pour nous retourner le bon contenu selon la langue choisie et selon la clé passée en paramètre. Ici citation est le nom de la clé dont on voudrait afficher la valeur

    // Do any additional setup after loading the view from its nib.
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

@end
```

Maintenant, exécutez votre projet en cliquant sur **Run**.
Vous pouvez télécharger le projet ici : Lien 11



Faites encore **Next** pour choisir d'enregistrer de votre projet sous votre Mac. Ça y est votre projet a bien été créé.

Retrouvez l'article de Salif Diop en ligne : Lien 12

Programmation fonctionnelle en Perl - Partie 1 : Les opérateurs de listes

Ce document est le premier d'une série de tutoriels visant à montrer comment utiliser certaines techniques de la programmation fonctionnelle en Perl et acquérir ainsi une bien meilleure expressivité. Cette première partie aborde en particulier les opérateurs de listes (grep, map, for, sort, etc.) et montre comment il est possible d'enchaîner certains d'entre eux pour former une espèce de « pipeline » dans lequel les données transitent et subissent une série de transformations successives permettant de résoudre simplement et élégamment des problèmes assez complexes.

1. Introduction

Ce tutoriel n'est pas destiné à enseigner la programmation Perl à des débutants, mais à introduire des techniques de programmation Perl relativement avancées qui supposent une assez bonne connaissance et une certaine expérience de la syntaxe de base de Perl. Les fonctions un peu avancées font l'objet de rappels de base permettant la compréhension des techniques employées, mais ne dispensent pas de consulter la documentation officielle.
Dans la préface de son excellent livre Higher Order Perl (HOP) (Lien 13), Mark-Jason Dominus remarque que les développeurs Perl tendent à « écrire du code C en Perl » (c'est-à-dire écrire du Perl comme si c'était du C). C'est une honte, ajoute-t-il, car Perl est bien plus expressif que le C. Nous pourrions faire bien mieux et utiliser Perl d'une façon dont les programmeurs C n'oseraient même pas rêver, mais nous ne le faisons pas la plupart du temps.
Perl possède de nombreux concepts et paradigmes avancés empruntés à la programmation fonctionnelle (notamment au langage Lisp) permettant bien souvent d'écrire facilement des programmes bien plus courts, bien plus expressifs, moins bogués et souvent plus lisibles que leurs équivalents en programmation procédurale ou orientée objet. En fait, Perl est sémantiquement bien plus proche du Lisp que du C, même si sa syntaxe de base est plus proche du C.
Je comptais au départ aborder dans ce tutoriel deux sujets principaux que l'on peut considérer comme des emprunts de Perl à Lisp : les « opérateurs de listes » et les « fonctions d'ordre supérieur ». Ce tutoriel s'avérant finalement plus long que prévu initialement, j'ai préféré le scinder en plusieurs parties, les deux sujets principaux pouvant être abordés de façon indépendante. Ce texte est donc le premier d'une série, consacrée aux opérateurs de listes.
Cette partie ne traite pas directement des concepts avancés de la programmation fonctionnelle proprement dite, mais elle rejoint cependant une caractéristique de beaucoup de langages fonctionnels, en particulier des différents dialectes de Lisp : la capacité d'appliquer une fonction à une liste ou un tableau, et, éventuellement de retourner une liste modifiée ou même une liste complètement différente.
Perl possède de très nombreux opérateurs de listes (le mot opérateur désigne ici une fonction prédéfinie dans le langage). Certains, bien connus, permettent d'ajouter ou de retirer un (ou plusieurs) élément(s) à un tableau (push, pop, shift, unshift, splice, etc.). Mais ce document s'intéresse surtout aux opérateurs qui permettent d'appliquer une transformation à tous les éléments d'une liste, d'utiliser une liste, de retourner une liste. Parmi ceux-ci, on peut citer map, grep, sort, join, split, print, printf, sprintf, die, exec, kill, system, warn, my, our, state, reverse, chomp, chop, scalar, keys, values, each, chmod, chown, unlink, utime, etc.
Le tableau suivant rappelle rapidement la fonction de ceux qui nous intéresseront le plus dans le présent tutoriel (ce tableau est un résumé très bref et ne constitue en aucun cas une documentation de référence, il convient de lire celle-ci pour les détails précis) :

Fonction ou opérateur	Syntaxe de base	Fonctionnalité
..	1..10	Renvoie une liste des nombres compris entre 1 et 10 (inclus).
join	join expr, liste	Renvoie la chaîne de caractères formée par l'insertion de expr entre les éléments de la liste et la concaténation du résultat.
split	split [motif [,expr]]	Utilise motif pour diviser expr (une chaîne) en une liste de chaînes. On peut également spécifier un nombre limite d'éléments renvoyés en résultat.
print	print [fh] liste	Imprime les éléments de liste. Si le descripteur de fichier fh est omis, imprime généralement sur la sortie standard (à l'écran).
reverse	reverse liste	Renvoie la liste en ordre inverse (en contexte de liste).
sort	sort [sous-routine] liste	Renvoie la liste triée. sous-routine doit renvoyer un nombre inférieur à 0, nul ou supérieur à 0 selon la façon dont deux éléments comparés

		doivent être ordonnés. Voir le chapitre sur sort.
foreach, for	for var (liste) bloc	Applique bloc à chaque élément de la liste. var (ou par défaut $_) est un alias sur les éléments successifs de la liste, les modifications sont faites sur les éléments de la liste. Voir ci-dessous.
grep	grep bloc liste	Évalue bloc pour chaque élément de la liste et renvoie (en contexte de liste) les éléments pour lesquels bloc a renvoyé vrai. Voir ci-dessous.
map	map bloc liste	Évalue bloc pour chaque élément de la liste et renvoie une liste d'éléments retournés par bloc. Voir ci-dessous.

Trois d'entre eux sont particulièrement puissants et méritent que l'on s'y attarde quelque peu :

- **for** (et son synonyme **foreach**) : cet opérateur prend une liste en entrée (à sa droite) et applique un bloc de code à chaque élément du tableau ou de la liste. Par exemple :
  À noter que la variable $item utilisée dans la boucle est un **alias** sur chaque élément successif du tableau ou de la liste ; en modifiant $item, on modifie bien le contenu du tableau ; si la variable $item n'est pas spécifiée, c'est la variable par défaut $_ qui sert d'alias sur les éléments du tableau ;

```
my @tableau = 1..10;    # @tableau contient 1, 2, 3, … 10
for my $item (@tableau) { $item *= 2;}; # @tableau contient 2, 4, 6 … 20
```

- **grep** : cet opérateur est un filtre : il prend une liste en entrée (à sa droite), applique un bloc de code à chaque élément et renvoie (en contexte de liste) les éléments pour lesquels ce bloc de code est évalué à vrai ; voici une autre manière d'alimenter un tableau avec les nombres pairs de 2 à 20, en filtrant les nombres impairs d'une liste de nombres de 1 à 20 :
  À noter que la variable spéciale $_ utilisée dans la boucle est un alias implicite sur chaque élément successif de la liste reçue en entrée et grep renvoie les éléments de la liste divisibles par 2 ; le nombre d'éléments renvoyés par un grep est inférieur ou égal au nombre d'éléments en entrée ;

```
my @tableau = grep { $_ % 2 == 0} 1..20;
```

- **map** : cet opérateur prend une liste en entrée (à sa droite), applique un bloc de code à chaque élément et retourne les nouveaux éléments ainsi calculés ; voici une troisième manière d'alimenter un tableau avec les nombres pairs de 2 à 20, en multipliant par 2 les éléments d'une liste de nombres de 1 à 10 :

```
my @tableau = map { $_ * 2 } 1..10;
```

Ici encore, la variable spéciale $_ utilisée dans la boucle est un alias implicite sur chaque élément successif de la liste reçue en entrée et map renvoie dans @tableau les éléments de la liste en entrée multipliés par 2 ; la fonction map peut retourner zéro, un ou plusieurs éléments pour chaque élément en entrée et peut donc renvoyer une liste plus longue ou plus courte que la liste en entrée. Par exemple, il est possible de renvoyer le double, le triple et le quadruple de chaque élément en entrée :

```
my @tableau = map { $_ * 2, $_ * 3, $_*4 } 1..5;
```

Ce qui donne la liste suivante dans @tableau : 2 3 4 4 6 8 6 9 12 8 12 16 10 15 20.

À noter que les fonctions grep et map admettent une autre syntaxe utilisant une expression et non un bloc de code (voir la documentation). Ainsi, l'exemple ci-dessus utilisant le grep pour générer une liste de nombres pairs peut être réécrit comme suit :

```
my @tableau = grep $_ % 2 == 0, 1..20;
```

Cette syntaxe nécessite une virgule pour séparer l'expression de la liste de valeurs. Mais cette autre syntaxe n'apporte rien à la présente discussion, je m'en tiendrai à la syntaxe avec un bloc de code présentée précédemment, d'usage bien plus général.

Les trois exemples donnés avec les fonctions grep et map enchaînent en fait deux opérateurs de listes : en commençant à lire par la droite, il y a d'abord l'opérateur « .. » (par exemple 1..10) qui génère une liste de nombres (de 1 à 10 dans l'exemple ; cette liste est ensuite fournie en entrée à l'opérateur map (ou grep) qui retravaille cette liste de départ pour en faire ce dont on a besoin. Cette technique est extrêmement puissante et mérite d'être explorée plus en détail, c'est l'objet principal de ce tutoriel.

2. Combiner les opérateurs de listes

La liste générée par le dernier map ci-dessus :

```
2 3 4 4 6 8 6 9 12 8 12 16 10 15 20,
```

est quelque peu désordonnée. Mais la magie des opérateurs de listes est que l'on peut les combiner pour faire des choses bien plus puissantes.

2.1. Enchaînements d'opérateurs

La fonction sort prend une liste ou un tableau en entrée et renvoie une liste triée. Utilisons-la sur le tableau généré précédemment :

```
my @tableau_trie = sort  @tableau;
```

Oups, ça ne marche pas bien, car cela donne la liste suivante :

```
10 12 12 15 16 2 20 3 4 4 6 6 8 8 9.
```

Par défaut, la fonction sort fait un tri de type lexicographique (ou, en simplifiant un peu, alphabétique), et non numérique : les nombres commençant par 1 viennent avant ceux commençant par 2 et ainsi de suite. Pour remédier à ce problème, il est possible d'utiliser la fonction map pour remplacer les nombres n'ayant qu'un seul chiffre par un nombre à deux chiffres commençant par 0 (remplacer par exemple « 3 » par « 03 »).

```
@tableau = map { length $_ == 1 ? "0$_": $_ }
@tableau;
```

Le map examine chaque élément du tableau à sa droite, il ajoute un zéro devant l'élément si c'est un nombre à un seul chiffre et renvoie vers le tableau à gauche de l'affectation une liste des éléments éventuellement modifiés. À noter que la variable @tableau se trouve à la fois en entrée (à droite) et en sortie (à gauche) de la fonction map. Perl gère très bien ce genre de situation. Le tableau contient maintenant :

```
02 03 04 04 06 08 06 09 12 08 12 16 10 15 20.
```

Il est maintenant possible de le trier correctement :

```
my @tableau_trie = sort  @tableau;
```

Ce qui donne la liste suivante :

```
02 03 04 04 06 06 08 08 09 10 12 12 15 16 20.
```

Mais tout ceci est bien laborieux. Il est possible de combiner les différentes opérations de listes pour générer directement le tableau trié, sans passer par des tableaux intermédiaires :

```
my @tableau_trie = sort map  { length $_ == 1 ?
"0$_": $_ }
                       map { $_ * 2, $_ * 3,
$_*4 } 1..5;
```

Ceci me donne la liste suivante dans @tableau_trie :

```
02 03 04 04 06 06 08 08 09 10 12 12 15 16 20.
```

Ici, nous utilisons un modèle de programmation par flux de données (dataflow programming), nous avons réalisé une espèce de pipeline de commandes successives sur les données (c'est un peu l'équivalent des pipes en programmation sous le shell Unix). Pour comprendre une commande comme celle-ci, il faut la lire de droite à gauche (et de bas en haut si le code est réparti sur plusieurs lignes). L'opérateur de liste « .. » avec les arguments 1 et 5 génère une liste de cinq éléments, les nombres de 1 à 5. Cette liste est passée en argument au map sur sa gauche, qui génère les doubles, les triples et les quadruples des nombres de 1 à 5. Ces quinze nombres sont passés au map plus à gauche, qui reformate les nombres n'ayant qu'un seul chiffre en une chaîne de caractères commençant par 0. La nouvelle liste ainsi produite est passée au sort qui trie les valeurs par ordre lexicographique. La liste triée résultante est renvoyée vers @tableau_trie.
Cette approche présente l'avantage de diviser un problème relativement complexe en une série de tâches plus simples. Cette stratégie s'appelle parfois « diviser pour régner » (divide and conquer). Les programmeurs Perl issus du monde de la programmation impérative procédurale classique mettent souvent pas mal de temps avant d'utiliser des fonctions comme grep et map. Le jour où ils décident de se l'approprier, ils se demandent comment ils ont pu s'en passer si longtemps.
Pour revenir à notre tâche, la liste obtenue n'est pas très satisfaisante : ces zéros au début des premiers nombres sont inélégants. Qu'à cela ne tienne ! Il suffit d'ajouter une nouvelle fonction map au pipeline pour éliminer ces zéros après le tri :

```
my @tableau_trie = map { s/^0//; $_ }
                   sort map { length $_ == 1 ?
"0$_": $_ }
                   map { $_ * 2, $_ * 3, $_*4 }
1..5;
```

Ce qui donne la liste suivante :

```
2 3 4 4 6 6 8 8 9 10 12 12 15 16 20.
```

Dans ce nouveau map, le bloc de code utilise le fait que map fait un alias de chaque élément de la liste qui lui est passée dans $_ et que l'opérateur de substitution s/// travaille par défaut sur cette même variable $_. La commande s/^0// retire donc le zéro initial de chaque élément de la liste commençant par un 0. Mais comme, par défaut, la commande s/// retourne non la variable modifiée, mais le nombre de substitutions effectuées, il faut réévaluer $_ modifié avant la fin du bloc. Dans les versions récentes de Perl (à partir de 5.14), il existe un modificateur s///r permettant de renvoyer la variable modifiée. Sur ces versions, le bloc map {s/^O//r ;} produirait le même résultat que le bloc map { s/^0// ; $_ } employé ci-dessus.
Nous reviendrons plus loin assez longuement sur cet enchaînement map {…} sort map {…} qui est extrêmement utile dans de nombreux cas.
Le tableau obtenu présente encore l'inconvénient d'avoir des doublons (double 4, double 6, etc.) que l'on pourrait désirer supprimer. Rien de plus simple, il suffit d'ajouter un grep pour filtrer les doublons :

```
my ($dupl, $prev) =  ("", "");
my @tableau_trie =  grep {$dupl = ($_ == $prev ?
0: 1); $prev = $_; $dupl}
                    map { s/^0//g; $_ }
                    sort map { length $_ == 1 ?
"0$_": $_ }
                    map { $_ * 2, $_ * 3, $_*4 }
1..5;
```

Mon tableau est maintenant dédoublonné : 2 3 4 6 8 9 10 12 15 16 20. Le fonctionnement du dédoublonnage est le suivant : le bloc grep compare chaque élément ($_) à son prédécesseur (stocké dans $prev) et retourne faux (0) à grep s'il est identique et vrai (1) s'il est différent, si bien que le grep l'élimine s'il est identique ; le bloc stocke également $_ dans la variable $prev pour pouvoir faire la comparaison lors de l'examen de la valeur suivante de la liste fournie en entrée. Mais si le mécanisme de détection des doublons vous échappe, peu importe (pour une première lecture), l'important ici est que ce grep ajouté

élimine les doublons de la liste.
À la réflexion, il n'y a pas vraiment besoin de la variable @tableau_trie si l'on désire juste afficher la liste à l'écran. Comme la fonction print est aussi un opérateur de liste, on peut juste ajouter un print devant le grep :

```
print grep {$dupl = ($_ == $prev ? 0: 1); $prev = $_; $dupl}
      map { s/^0//g; $_ }
      sort map { length $_ == 1 ? "0$_": $_ }
      map { $_ * 2, $_ * 3, $_*4 } 1..5;
```

Ce qui imprime : 2346891012151620. Ah, ce n'est pas l'affichage recherché, il faudrait des espaces ou des tirets entre les différentes valeurs pour les différencier. Ajoutons une fonction join qui va construire une chaîne de caractères avec les éléments de la liste séparés par des tirets :

```
print join '-', grep { $dupl = ($_ == $prev ? 0: 1); $prev = $_; $dupl}
                map { s/^0//g; $_ }
                sort map { length $_ == 1 ? "0$_": $_ }
                map { $_ * 2, $_ * 3, $_*4 } 1..5;
```

Ce qui imprime à l'écran la liste de valeurs séparées par des tirets :

```
2-3-4-6-8-9-10-12-15-16-20.
```

Pour récapituler, on enchaîne maintenant dans le pipeline huit opérateurs de listes, dont six différents. Bon, j'ai peut-être un peu exagéré, il est assez rare de faire des constructions aussi complexes, mais j'espère avoir montré comment il était facile d'ajouter de nouvelles étapes de traitement dans ce genre de pipeline de données. On est très loin du C, non ?

Le genre de pipeline de données décrit ci-dessus est très efficace et très utile, mais il convient, comme pour beaucoup de bonnes choses, de ne pas en abuser et de veiller à ne pas obscurcir le code. Les exemples ci-dessus enchaînent jusqu'à huit opérateurs de listes (ce qui est déjà beaucoup), mais chaque opérateur ne faisant qu'une seule chose, il reste facile de comprendre pas à pas la succession des actions appliquées aux données en entrée. La mise en page du code en plusieurs lignes peut contribuer à clarifier l'enchaînement des tâches. Il ne faut pas hésiter, le cas échéant, à bien commenter le code, par exemple en donnant un échantillon des données en entrée et en sortie.

2.2. Retour sur la fonction sort

Faisons maintenant un aveu et révélons un point qui avait été provisoirement gardé sous le silence. Une partie de ce qui a été fait ci-dessus était en fait complètement inutile ; le but était purement pédagogique : montrer l'enchaînement naturel des traitements dans le pipeline. La fonction sort est fort heureusement tout à fait capable de trier une liste numériquement, sans avoir besoin de reformater les données comme nous l'avons fait ci-dessus. Il suffit de lui passer une fonction ou un bloc de code explicitant une comparaison numérique (et non lexicographique ou alphabétique). L'ajout du 0 initial et son retrait ultérieurement étaient en fait inutiles.
La fonction sort peut prendre un bloc de code, comme les fonctions map et grep, pour déterminer la façon dont sont faites les comparaisons entre les éléments du tableau à trier. En écrivant ceci :

```
my @tableau_trie = sort { $a <=> $b }
                   map { $_ * 2, $_ * 3, $_*4 }
1..5;
```

on obtient directement un tableau trié contenant :

```
2 3 4 4 6 6 8 8 9 10 12 12 15 16 20.
```

Une fonction de tri doit généralement effectuer des comparaisons d'éléments deux à deux. La fonction sort de Perl n'échappe pas à cette règle, mais c'est à l'utilisateur de spécifier comment cette comparaison doit être effectuée. Le bloc {$a <=> $b} dit ceci : quand on compare deux éléments $a et $b, ce bloc retourne -1 si $a doit précéder $b dans la liste triée finale, 1 si $b doit précéder $a et 0 si les valeurs sont identiques. La fonction sort utilise les valeurs positive, négative ou nulle renvoyées par le bloc pour déterminer l'ordre dans lequel classer les éléments.
Nous pouvons donc simplifier le pipeline de commandes et supprimer les deux appels de map chargés du reformatage en modifiant le sort :

```
print join '-', grep { $a = $b == $_? 0: 1; $b = $_; $a;}
                sort {$a <=> $b}
                map { $_ * 2, $_ * 3, $_*4 }
1..5 ;
```

2.3. Un minimum de réflexion ne nuit pas

En réalité, cette série de commandes est encore ridiculement compliquée, compte tenu des données en résultant. Le seul but était de montrer comment enchaîner une série d'opérateurs de listes pour manipuler un flux de données. Quel est le résultat final de cette série de commandes ? Une liste triée des multiples de 2, de 3 et de 4 inférieurs ou égaux à 20. Comme les multiples de 4 sont aussi des multiples de 2, c'est en fait la liste triée des multiples de 2 et de 3 inférieurs ou égaux à 20.
Vous vous souvenez comment nous avons généré au tout début une série de nombres pairs de 0 à 20 avec la fonction grep ? Nous avons généré une liste de nombres de 1 à 20 et utilisé un filtre (divisible par 2) pour ne retenir que les nombres pairs. Il suffit ici de faire la même chose en utilisant deux filtres successifs : divisible par 2 ou divisible par 3 :

```
print join "-", grep { $_ % 2 == 0 || $_ % 3 == 0 } 1..20 ;
```

Cette commande imprime bien la liste de nombres suivants, séparés par des tirets :

```
2-3-4-6-8-9-10-12-14-15-16-18-20.
```

Le bloc de code dans le grep évalue d'abord si le nombre est pair et renvoie vrai si c'est le cas (l'élément est alors passé à la commande suivante, le join); si le nombre n'est pas pair, alors le bloc évalue si le nombre est divisible par 3 ; si c'est le cas, l'élément est passé à la suite du

traitement ; si l'élément n'est ni pair ni divisible par 3, alors il est éliminé de la liste de sortie.
Cette suite de commandes est beaucoup plus simple que celle que nous avions vue antérieurement. Plus de sort, plus de map, seulement trois opérateurs de listes. Cela montre que, si le but n'avait pas été d'illustrer pas à pas la construction progressive d'un pipeline de données, j'aurais sans doute dû réfléchir un peu plus au résultat recherché avant de me lancer aveuglément dans un enchaînement beaucoup trop compliqué de commandes. La morale est que l'utilisation de cette technique de programmation des opérateurs de listes enchaînés permet des constructions extrêmement puissantes, mais ne dispense pas d'un minimum de réflexion.
En fait, il est possible de réduire un peu plus le nombre de caractères de la série de commandes en inversant le test de parité et de divisibilité par 3 :

```
print join "-", grep {not $_ % 2 && $_ % 3 }
1..20;
```

Mais le code résultat est sans doute nettement moins lisible ; la version précédente, qui dit plus clairement ce qu'elle fait, est certainement bien préférable.

2.4. Petite digression sur la question des performances

La simplicité du code est un objectif capital, mais, parfois, d'autres critères sont tout aussi cruciaux, voire plus, par exemple la rapidité d'exécution. On peut avoir besoin de créer une structure de données plus complexe pour améliorer les performances. Dans certains cas, la plupart des données en entrée ne sont pas utiles pour ce que l'on désire extraire ; filtrer dès le départ les données en entrée et garder les seules données utiles peut faire gagner beaucoup de temps. Je dois souvent, dans le cadre de ma profession, retraiter des fichiers de plusieurs dizaines ou même centaines de millions de lignes. Assez souvent, la première chose à faire est d'instaurer un filtre qui éliminera peut-être 90 % ou même 98 % des lignes et permettra de ne retraiter que les 10 % ou 2 % utiles. Le gain de temps peut dans certains cas être considérable.
Modifions un peu la recherche précédente de nombres divisibles par 2 et 3, en spécifiant que l'on désire cette fois garder les nombres entre 1 et 50 qui sont à la fois divisibles par 2 et par 3 (et non plus divisibles par 2 ou par 3). Ceci donne par exemple le code suivant :

```
print join "-",
      grep { $_ % 2 == 0 && $_ % 3 == 0 }
      1..50; # noter le && au lieu du ||
```

Ce qui affiche le résultat suivant : 6-12-18-24-30-36-42-48
Mais les nombres divisibles par 2 et par 3 sont les nombres divisibles par 6. Il est donc plus simple d'écrire :

```
print join "-", grep { $_ % 6 == 0 } 1..50; #
affiche la même chose
```

Pour les nombres inférieurs à 50, le résultat est instantané, les performances n'ont aucune importance, on choisira la solution la plus simple ou la plus claire.
Mais si l'on désire les nombres inférieurs à dix millions, les performances ont peut-être de l'importance. Laquelle des deux solutions est-elle la plus rapide ? La seconde ne fait qu'une opération là ou la première en fait deux, mais, d'un autre côté, la division par 2 est peut-être très rapide en binaire et filtre d'entrée de jeu la moitié des nombres. Hum, difficile à dire. Faisons un petit benchmark rudimentaire. Sous Unix, la fonction time donne directement la durée d'exécution d'un programme. Essayons des scripts unilignes :

```
$ time  perl -e '@c=  grep { $_ % 6 == 0 }
1..10000000; print $#c +1;'
1666666
real    0m2.837s
user    0m2.262s
sys     0m0.155s


$ time  perl -e '@c=  grep { $_ % 2 == 0 && $_ %
3 == 0 } 1..10000000; print $#c +1;'
1666666
real    0m2.987s
user    0m2.854s
sys     0m0.139s
```

La solution avec la divisibilité par 6 paraît légèrement meilleure, mais la différence est trop faible pour en être certain (et même trop faible pour qu'il soit vraiment utile de se poser la question). Il existe des modules Perl de benchmark bien plus avancés et plus précis (ils donnent un avantage un peu plus net au test de divisibilité par 6), mais il n'est pas utile de sortir ici de l'artillerie lourde pour chasser une simple mouche.
Peut-être que tester si le dernier chiffre est pair est plus rapide ? Voyons voir :

```
$ time  perl -e '@c=  grep { /[02468]$/ && $_ % 3
== 0 } 1..10000000;
                      print $#c +1;'
1666666
real    0m15.215s
user    0m14.975s
sys     0m0.233s
```

Non, pas du tout, tester la parité du dernier chiffre avec une expression régulière donne un bien moins bon résultat. (La précompilation de l'expression régulière ou l'utilisation de l'option /o n'améliorent pas les choses.)
Et si l'on commençait par un map multipliant par 2 des nombres de 1 à 5 millions pour ne générer que des nombres pairs inférieurs à 10 millions et testait ensuite la seule divisibilité par 3 ?

```
$ time  perl -e '@c=  grep { $_ % 3 == 0 }
                      map { $_* 2} 1..5000000;
print $#c +1;'
1666666
real    0m2.106s
user    0m2.058s
sys     0m0.046s
```

Ah, surprise (ou pas tant que cela, à vrai dire je m'y attendais un peu, mais je n'aurais pas parié dessus avant de tester), cette solution est meilleure que les précédentes : enchaîner le map (*2) et le grep (divisibilité par 3) donne des temps 35 % meilleurs que le grep simple (divisibilité par 6). Mais, au fait, n'ai-je pas manqué de réflexion en faisant les opérations dans cet ordre ? Ne vaut-il pas mieux commencer par filtrer les multiples de 3 puis multiplier par 2 ? Voyons cela :

```
$  time perl -e '@c= map { $_* 2}  grep { $_ % 3
== 0 } 1..5000000; print $#c +1 ;'
1666666
real    0m1.539s
user    0m1.450s
sys     0m0.077s
```

Nous obtenons un nouveau gain significatif. Nous laisserons au lecteur le soin de tester une dernière solution bien plus simple :

```
$  time perl -e '@c= map { $_* 6}  1..1666666;
print $#c +1 ;'
```

Pour dire la vérité, les performances ne sont pas le but principal des pipelines de données. Dans un cas général, une boucle foreach sera assez souvent plus rapide, car elle ne copie pas les données plusieurs fois (de plus, dans certaines circonstances, il est possible d'arrêter une boucle foreach dès qu'elle a fourni le résultat désiré, alors que ce n'est pas possible avec les opérateurs de type map ou grep). Le but principal est de simplifier la programmation (une heure d'un développeur coûte bien plus cher qu'une heure de CPU). Mais il arrive assez fréquemment que le pipeline permette d'améliorer (parfois considérablement) les performances.

Dans le cas présent, entre 2,8 secondes et 2,1 secondes ou même 1,5 seconde, peu importe cette différence, mais il est intéressant de se poser la question et de réfléchir aux résultats et à leurs conséquences. Ici, la solution initialement la plus simple (un seul grep) n'était pas la meilleure, un map suivi d'un grep font assez nettement mieux. Finalement, un map différent fait encore mieux. Encore une fois, dans le cas de l'exemple, la solution la plus simple était bien suffisante, mais il est des situations où ajouter un peu de complexité apparente permet au programme (et à l'ordinateur) de mieux travailler. Parfois beaucoup mieux. C'est ce que nous allons voir avec certains algorithmes de tri.

Retrouvez la suite de l'article de Laurent Rosenfeld en ligne : Lien 14

Petit guide du templating client

Avez-vous déjà manipulé de grands pans de HTML en JavaScript, et trouvé ça peu lisible et fastidieux ? Les templates sont là pour vous simplifier la vie et produire du code plus lisible et maintenable. Cet article fait le tour des principales solutions de templating existantes en JavaScript pour vous permettre de trouver la bibliothèque qu'il vous faut.

1. Introduction

Les navigateurs ne cessent de s'améliorer et les applications Web mettent de plus en plus à profit le gain de performance des moteurs d'exécution JavaScript ainsi que les nouvelles API offertes par les évolutions des langages côté client. Le requêtage dynamique permis par AJAX (Lien 15) fut à lui seul une petite révolution pour les sites Internet, ouvrant la porte à des sites plus interactifs, plus communicants. Permettant de récupérer des données serveur à la demande sans changer de page et sans recharger les éléments communs du site tels que sa structure HTML (Lien 16), ses bibliothèques JavaScript et ses feuilles de style CSS (Lien 17), AJAX fut une composante technologique essentielle pour permettre l'essor du Web moderne représenté par les réseaux sociaux, les webmails et autres « Rich Internet Applications ».
Il se trouve également être à un virage dans l'histoire de l'innovation des technologies Web. En effet, les années 1995 à 2005 ont été marquées par la « bulle Internet » et le « Web 2.0 ». Mais la plupart des innovations de cette décennie résidaient dans les technologies de gestion, de stockage et de génération de contenu employées côté serveur. La première version publique de PHP (Lien 18) est apparue en 1995, tandis que JavaEE (Lien 19) et ASP.NET (Lien 20) sont apparues au début du millénaire pour répondre aux nouveaux usages et besoins des sites de publication et d'échange. L'outillage côté serveur s'étant considérablement amélioré, il s'agissait dès lors de se pencher du côté client.
Emporté par la vague d'AJAX, c'est tout le développement Web front-end qui bénéficia d'un essor depuis. La préparation de la norme HTML5, la révolution mobile et les applications Web multicanaux ont également concouru à cet engouement autour du navigateur. Le rythme de développement de tous les navigateurs s'est accéléré tout comme l'apparition de nouvelles API, bibliothèques et autres outils de développement côté client. De nouveaux modèles de conception suivant cette tendance voient le jour, tels que les Single Page Applications qui sont de plus en plus populaires. Le principe est de ne charger qu'une seule page HTML (souvent statique) depuis le serveur, puis de reposer intégralement sur des requêtes AJAX pour la navigation et l'interaction. L'avantage est que l'on peut réduire le nombre et la taille de ces requêtes, en se contentant des données dans leur expression la plus simple, c'est-à-dire dans un format sérialisé comme JSON ou XML (Lien 21).
Toutes ces nouveautés de plus en plus incontournables ont pour première conséquence de délocaliser toujours plus d'intelligence applicative côté JavaScript. Les développeurs ont alors dû s'adapter et inventer de nouveaux outils pour manipuler plus facilement le Document Object Model (DOM) et interagir avec la page en JavaScript. Vous connaissez probablement tous jQuery, la bibliothèque JavaScript la plus populaire qui s'est fait connaître pour sa fonction sélecteur quasi omnipotente. Aujourd'hui, je vous propose de passer au niveau supérieur en vous présentant tout ce qu'il y a à savoir sur le templating côté client.

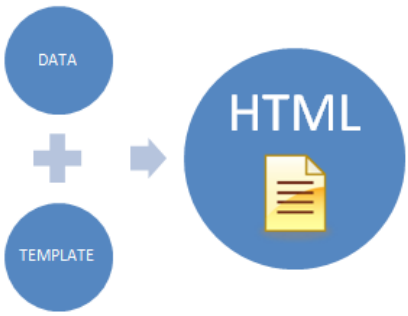
2. Qu'est-ce que le templating ?

Les templates sont des modèles décrivant la manière dont les données sont composées pour former l'aspect final d'un document ou d'une partie d'un document. Dans le cas du Web, ils serviront donc à produire de manière descriptive et intuitive le HTML qui composera vos pages. Historiquement, les outils de templating travaillaient presque tous côté serveur. Certains mêmes étaient des composantes intégrales d'un langage. Ainsi vous connaissez probablement tous les balises <? ?> de PHP ou encore <% %> de JSP (Lien 22).

Exemple PHP
```
<h3>Auteur(s) du livre:</h3>
<ul>
<?
    foreach ($authors as $name) {
        echo "<li>$name</li>";
    }
?>
</ul>
```

Exemple JSP
```
<h3>Auteur(s) du livre:</h3>
<ul>
<%
    for (int i = 0; i < authors.length; ++i) {
%>
        <li><%= authors[i] %></li>
<%
    }
%>
</ul>
```

Tous les outils de templating s'utilisent à peu près de la même façon : on envoie le template d'un côté, les données de l'autre, et il en sort le HTML final de la vue.



3. Pourquoi délocaliser le templating côté client ?

Si j'ai parlé d'AJAX en introduction, c'est qu'il est une des raisons principales pour laquelle déplacer ce travail de templating du serveur vers le client peut être une bonne idée. Rien n'empêche de faire une requête AJAX et d'utiliser un templating côté serveur pour générer un fragment de HTML, puis l'envoyer au client pour l'injecter dynamiquement dans la page en JavaScript. Seulement le dynamisme apporté par AJAX est modéré par la durée de la requête, qui comprend la latence réseau et le temps de traitement. De plus, il se peut que certaines actions sur votre site entraînent une modification de la vue sans pour autant que notifier le serveur soit nécessaire. On se retrouve ainsi à devoir faire appel à JavaScript de temps à autre pour modifier la page. Dès lors, pourquoi ne pas franchir le pas et confier tout le templating à JavaScript ?

Modèle de données

```
var data = {
    title : "Good Omens",
    authors : [ "Terry Pratchett", "Neil
Gaiman" ]
};
```

Template (bibliothèque EJS)

```
<h2><%= title %></h2>
<h3>Auteur(s) du livre:</h3>
<ul>
<% for(var i=0; i<authors.length; i++) { %>
    <li><%= authors[i] %></li>
<% } %>
</ul>
```

Résultat

```
<h2>Good Omens</h2>
<h3>Auteur(s) du livre:</h3>
<ul>
    <li>Terry Pratchett</li>
    <li>Neil Gaiman</li>
</ul>
```

Le code ci-dessus est un template EJS (Embedded JavaScript). Il s'agit d'une bibliothèque JavaScript, d'à peine 10 KB, qui s'occupe de lire et d'interpréter les templates tels que celui-ci.

Hé, ce n'est pas très différent du templating serveur.

Hé non ! C'est tout dit la même chose. J'ai choisi de vous montrer EJS, car sa syntaxe est volontairement destinée à ne pas brusquer les développeurs habitués à JSP et PHP. La différence majeure réside non pas dans la déclaration des templates, mais dans leur utilisation. Côté serveur, les outils de templating prennent naturellement la main au moment de générer des réponses aux requêtes HTTP reçues. Côté client par contre, vous avez le total contrôle de quand et comment faire appel à votre bibliothèque de templating.
Si vous n'êtes pas encore décidé, voilà un tableau synthétique des avantages et inconvénients du templating côté serveur ou côté client :

Côté serveur	Côté client
Avantages	
Votre site peut toujours fonctionner sur les navigateurs avec JavaScript désactivé (moins de 1 % des requêtes d'après les statistiques de Yahoo en 2011). La logique de composition de vos vues n'est pas transmise au client. Si vous avez des choses à cacher, elles seront plus difficiles à trouver de l'extérieur.	Les requêtes peuvent être allégées en passant les données sous leur forme la plus simple « only Data on Wire ». Vous bénéficiez d'un contrôle accru de quand et comment vos templates sont utilisés. Votre templating peut toujours fonctionner lorsque la connexion est perdue.
Inconvénients	
Le rendu des templates consomme des ressources serveur (généralement inférieures à celles utilisées par les requêtes BDD). Chaque utilisation d'un template implique une requête HTTP et un aller-retour client-serveur.	Certains navigateurs particulièrement obsolètes peuvent ne pas supporter la bibliothèque de templating (le support reste généralement très bon). Le temps de rendu peut varier selon le terminal client (bien que généralement très court et parfaitement négligeable).

4. Le faux argument de la performance

Les progrès des navigateurs et des moteurs d'exécution JavaScript tendent à naturellement faire pencher la balance du côté client si l'on réfléchit en termes de performance pure. En effet, même si les templating serveurs resteront au niveau matériel et logiciel plus performants que leur équivalent client, le temps de rendu des templates reste largement inférieur à la latence réseau. Prenons un exemple concret : générer une liste de 10 000 livres et leurs auteurs.
Format HTML attendu :

```
<ol>
    <li>
        <h2>David Copperfield</h2>
        <h3>Auteur(s) du livre:</h3>
        <ul>
            <li>Charles Dickens</li>
        </ul>
    </li>
    ...
</ol>
```

Format sérialisé JSON :

```
{"books":[{"title":"David Copperfield","authors":
["CharlesDickens"]},...]}
```

Admettons que les informations pour un livre occupent en moyenne 120 caractères sous leur forme HTML et 60 sous leur forme JSON. La liste complète de 10 000 livres occupe donc 1,2 Mo en HTML et 600 ko sous forme JSON.
Après benchmark sur un smartphone de gamme moyenne, la bibliothèque de templating JavaScript la plus performante testée (Hogan.js) donne 200 ms de temps de rendu pour la liste de 10 000 éléments. Admettons que l'on dispose d'un serveur puissant ne croulant pas sous la charge et capable de générer cette même liste en deux fois moins de temps, grâce au matériel supérieur et à la précompilation des templates. Cela nous donne donc 100 ms de gagnées en temps de rendu du template en dépit de 600 ko de plus dans la requête. En supposant que ce même smartphone soit dans de bonnes conditions réseau et reçoive la 3G à un débit moyen de 5 Mb/s, les 600 ko de plus prendront donc environ… une seconde de plus à arriver. Même avec la vitesse des réseaux actuels, le temps de rendu s'avère dix fois plus court que le temps de requête.
Cet exemple est très approximatif et hypothétique, mais il sert juste à rejeter le faux argument exposé au premier abord par les réfractaires du templating client-side selon qui le templating JavaScript n'est pas assez performant par rapport aux solutions serveur.
J'espère à présent vous avoir convaincu que déplacer le templating côté client présente un réel intérêt, et vous dispensera à l'avenir de « code spaghetti » pour modifier vos vues localement.

5. Tour d'horizon des bibliothèques de templating

Quand il s'agit de concevoir un outil de templating, tout le monde n'est pas d'accord sur la part de logique à mettre du côté vue (HTML) et la part de logique à mettre du côté modèle (JavaScript). Certains préfèrent la simplicité en utilisant des syntaxes très permissives voire en écrivant du code JS au sein du HTML. D'autres préfèrent les templates dits « logic-less », afin d'épurer le code des vues et de s'assurer une plus grande proximité entre le modèle de données et ce qui est affiché sur la page.
Question lisibilité, il est difficile de s'accorder. L'un rendra la logique de la vue plus explicite, l'autre mettra le HTML final de la vue en évidence. Il convient donc de choisir selon vos préférences et vos besoins. Néanmoins, la tendance générale est davantage tournée vers les templates logic-less comme le montrent les choix opérés pour Angular et Ember, deux des frameworks JavaScript les plus populaires du moment.
Si on trace un axe allant du plus au moins en quantité de logique côté template, et que l'on y place quelques outils de templating connus, voilà ce que ça donnerait (attention, ce graphe est issu de mon interprétation personnelle des choix faits pour chaque bibliothèque)

Nous n'allons pas tous les passer en revue. J'en ai sélectionné quatre qui représentent bien selon moi les différentes solutions existantes et leurs différences. Pour chacune, nous allons reprendre l'exemple de la liste de livres et comparer les implémentations. Voici les données que nous utiliserons :

Modèle de données

```
var data = [
    {
        title : "David Copperfield",
        authors : [ "Charles Dickens" ]
    },
    {
        title : "Romeo and Juliet",
        authors : [ "William Shakespeare" ]
    },
    {
        title : "Good Omens",
        authors : [ "Terry Pratchett", "Neil
Gaiman" ]
    }
];
```

5.1. La vieille méthode : innerHTML ou jQuery.append

```
var output = "<ol>";

for(var i=0; i < data.length; i++){
    var book = data[i];
    output += "<li><h2>" + book.title + "</h2>"
            + "<h3>Auteur"+(book.authors.length>1
? "s" : "") + "du livre:</h3>"
            +"<ul>";
    for(var j=0; j< book.authors.length; j++){
        output += "<li>" + book.authors[j] +
"</li>";
    }
    output += "</ul>";
}

output+="</ol>";
document.getElementById("result").innerHTML =
output;
```

Presque tous les codeurs en JavaScript sont déjà passés par là. On construit simplement le HTML sous forme de String en y concaténant les données aux endroits désirés, avant de l'injecter dans le DOM avec la propriété innerHTML du conteneur à remplir. Il n'y a pas de template, le code logique est mélangé au code de la vue. Cette méthode est de loin la plus flexible, l'intégralité de la vue étant gérée au caractère près par votre code JavaScript. Mais c'est également la moins lisible et la moins maintenable, ce qui apporte vite son lot de difficultés pour les projets et les vues plus complexes. Heureusement, on a inventé les templates !

5.2. DOMjs

Lien 23

Plutôt que de songer à créer une syntaxe spécifique pour les templates, certains préfèrent simplement déclarer de nouvelles fonctions pour rendre le code ci-dessus plus lisible. L'idée qu'a eue le créateur de DOMjs est de déclarer une fonction JavaScript pour chaque tag HTML permettant de générer l'élément HTML associé et son

contenu à la demande. **DOMjs est une bibliothèque en cours de développement, incomplète, mal documentée et certainement pas prête pour être utilisée en production**, mais c'est un bon exemple de solution intermédiaire que l'on peut imaginer avant de faire appel à du vrai templating :

```
var template = function(){

    var output = ol();
    for(var i=0; i < data.length; i++){
        var book = data[i];
        var authorsList = ul();

        for(var j=0; j< book.authors.length; j++)
{
            authorsList( li(book.authors[j]) );
        }

        output(
            li(
                h2(book.title),
                h3("Auteur"+
(book.authors.length>1 ? "s" : "")+"du livre:"),
                authorsList
            )
        );
    }

};

document.getElementById("result").appendChild(dom
js.build(template));
```

Le code est finalement assez proche de celui utilisant innerHTML mais en plus allégé. La différence majeure, cependant, réside dans le fait que l'on manipule des DOMElement plutôt que des String. On a donc la possibilité d'utiliser toutes les méthodes, propriétés et évènements JavaScript à notre disposition tels que setAttribute, onclick, dataset… Autrement dit une syntaxe plus riche et plus légère. Mais nous avons toujours du HTML-like perdu au milieu de notre JavaScript. Tâchons d'aller plus loin.
Dans le même genre :

- Jade : Lien 24

5.3. John Resig micro templating

Lien 25

Cette bibliothèque microscopique (moins de 1 ko) est un petit bijou de John Resig, que certains connaissent déjà pour être le créateur de jQuery. Il fournit simplement une fonction tmpl prenant en paramètre l'ID d'un template, et les données sous forme d'objet JavaScript.

```
<script type="text/html" id="books_template">
    <ol>
        <% for ( var i = 0; i < data.length; i+
+ ) { %>
        <li>
            <h2><%=data[i].title%></h2>
            <h3>Auteur<%
if(data[i].authors.length > 1){%>s<%}%> du
livre:</h3>
            <ul>
                <% for ( var j = 0; j <
data[i].authors.length; j++ ) { %>
                <li><%=data[i].authors[j]%></li>
                <% } %>
            </ul>
        </li>
        <% } %>
    </ol>
</script>
```

```
document.getElementById("result").innerHTML =
tmpl("books_template", data);
```

Le template est défini par une balise script qui peut être placée n'importe où dans la page (mais pas dans le conteneur « result » dont le contenu va être écrasé avec innerHTML). Notez l'attribut type de la balise script. Il ne s'agit pas d'un attribut standard, simplement les navigateurs n'essaieront pas de l'interpréter comme du JavaScript puisqu'ils ne reconnaîtront pas le type.
La syntaxe des templates ne devrait pas déconcerter les développeurs habitués aux JSP. L'avantage de cette bibliothèque outre sa petite taille est la facilité avec laquelle on peut intercaler du JavaScript entre les nœuds HTML, apportant lisibilité et flexibilité à la logique de la vue. Cependant, il peut être difficile de discerner rapidement le HTML final rendu par le template. Et si on tâchait de réduire la quantité de logique côté template ?
Dans le même genre :

- Embedded JavaScript (EJS) : Lien 26
- doT: Lien 27

5.4. mustache

Lien 28

mustache est la première syntaxe de templating abordée dans cet article que l'on pourrait qualifier de « logic-less ». En effet, seules des fonctions basiques comme les conditions ou les boucles sont permises au sein du template. Tout autre traitement plus complexe devra être fait au préalable en JavaScript avant d'envoyer les données au moteur de templating : c'est ce qu'on appelle le **data preprocessing**. Ce sont toutes les opérations qui vont venir s'intercaler entre le modèle de données initial et la vue. Dans notre exemple, on peut par exemple préprocesser la règle pluriel appliquée au label « Auteur(s) du livre ».
La syntaxe mustache est utilisée dans un grand nombre de langages différents. C'est pour ainsi dire une des références en matière de templating logic less, c'est pourquoi vous serez amenés à la retrouver dans beaucoup de bibliothèques de templating JS telles que handlebars, ICanHaz, ou avec quelques différences/ajouts comme pour Hogan ou Dust. Pour l'exemple, j'utiliserai ICanHaz qui permet d'externaliser vos templates facilement dans des balises script au sein de votre page.
Voilà ce cela que donne :

```
<div id="result"></div>
<script id="books_template" type="text/html">
    <ol>
    {{#books}}
        <li>
            <h2>{{title}}</h2>
            <h3>Auteur{{#pluriel}}s{{/pluriel}}
du livre:</h3>
```

```
            <ul>
            {{#authors}}
                <li>{{.}}</li>
            {{/authors}}
            </ul>
        </li>
  </ol>
</script>
```

```
//Data preprocessing
for(var i=0; i < data.length; i++){
    var book = data[i];
    book.pluriel = (book.authors.length > 1);
}
//Template rendering
var html = ich.books_template({ books: data });

//HTML appending
document.getElementById("result").innerHTML =
html;
```

Vous constaterez que le template est tout de suite beaucoup plus lisible. Néanmoins il a fallu d'abord préprocesser les données . Si dans cet exemple, c'est assez trivial, il arrive que cette étape soit beaucoup plus fastidieuse dans des templates plus complexes. Il revient à vous de décider si ce traitement est fait côté serveur ou côté client, et comment l'isoler du reste de votre code. Gardez toutefois à l'esprit qu'il peut être utile de disposer des mêmes données dans son modèle et sur la page, telles que les voient vos utilisateurs. Toute la logique que vous mettrez dans vos templates ne pourra pas être réutilisée ailleurs dans votre application. C'est pourquoi l'utilisation d'un template logic less peut au début vous apparaître comme une contrainte, mais finalement vous obliger à modulariser votre code et éviter les copier/coller : un bon point pour votre application !
Dans le même genre :

- Handlebars : Lien 29
- Dust : Lien 30
- Hogan : Lien 31

5. 5. PURE

Lien 32

Terminons par Pure qui comme son nom l'indique cherche à épurer au maximum les templates. Contrairement à JRMT et Mustache, il s'agit d'un templating basé sur le DOM : on associe les données à des éléments HTML et non en manipulant des String pour former le HTML final. Dans le cas de PURE, cette association de données aux nœuds HTML peut se faire de deux façons : soit simplement en utilisant l'attribut class ; soit en utilisant des sélecteurs plus complets pour décrire les liens données-DOM dans une hashmap appelée directive. Regardons le code, c'est bien plus parlant :

```
<div id="result">
    <ol>
        <li class="books">
            <h2 class="title"></h2>
            <h3 class="authorsLabel"></h3>
            <ul>
                <li class="authors"></li>
            </ul>
        </li>
    </ol>
</div>
```

```
//Data preprocessing
for(var i=0; i < data.length; i++){
    var book = data[i];
    book.authorsLabel = "Auteur" +
(book.authors.length > 1 ? "s" : "") + " du
livre";
}

$p("#result").render(data);
```

Utiliser les classes pour définir les associations data-HTML nous oblige à ajouter des classes aux éléments HTML. Il y a un gros avantage et un gros inconvénient à cela. L'avantage, c'est que cela leur donne un sens explicite et facilite considérablement la lecture du DOM et des règles CSS qui lui sont associées. L'inconvénient, c'est que cela pose un risque de conflit avec des règles CSS existantes en mélangeant le templating avec la gestion du style. Ce qui fait que ce système de templating est plus intrusif que les autres, ce qui peut être gênant notamment lorsque l'intégrateur n'est pas la personne qui a fourni la maquette HTML et le design. C'est pourquoi l'auteur a imaginé le système de directives pour gérer les associations côté JavaScript et permettre au développeur d'utiliser les sélecteurs de son choix plutôt que d'imposer l'utilisation de classes.

```
var directive = {
    'li.books': 'books',
    'h2': 'title',
    'h3': 'authorsLabel',
    'li.authors': 'authors'
};

$p("#result").render(data, directive);
```

Dans le même genre :

- Transparency: Lien 33

6. Quel templating choisir ?

Avant tout, si j'ai tenu à vous présenter quatre systèmes de templating très différents, c'est pour que vous puissiez vous forger votre propre opinion et choisir en toute connaissance de cause. Chaque approche a ses avantages et ses inconvénients. Mon choix personnel ? J'aime les templates très épurés et je crois beaucoup aux bénéfices du logic less et du templating DOM-based. Mais je n'ai pas trouvé d'outil de templating qui m'apporte totale satisfaction. Alors j'ai comme projet dans les cartons de développer ma propre bibliothèque de templating basée sur les data-attributes : logic-less, DOM-based, non intrusive. La syntaxe de KnockoutJS me plaît beaucoup, dommage que je ne puisse pas utiliser uniquement la partie templating de ce framework MVVM.
Je vous recommande de tester rapidement plusieurs bibliothèques qui vous plaisent au premier abord avant de faire votre choix. Et si vous hésitez encore, il existe ce site : Lien 34 qui vous permettra de trouver le système de templating qu'il vous faut en répondant à quelques questions.

7. Aller plus loin

7.1. Le data-binding

Vous connaissez AngularJS, Ember, KnockoutJS ? Tous ces frameworks JavaScript assez populaires en ce moment proposent quelque chose d'un peu plus sophistiqué que du templating : ils peuvent générer facilement du HTML avec des données JavaScript, mais aussi modifier ces données directement en fonction de la saisie de l'utilisateur dans le HTML ou encore mettre à jour automatiquement la vue lorsque les données changent côté modèle. Cette sorte de templating à double sens est couramment appelée **data-binding**.
On pourrait certainement considérer le data-binding comme l'évolution du templating. Et pour être utilisateur régulier d'AngularJS, j'ai pleinement conscience de tout le potentiel du data-binding. Seulement il n'est pas adapté à toutes les applications, mais plutôt à celles qui fonctionnent beaucoup sur le temps réel avec des mises à jour serveur et des saisies utilisateur fréquentes. Pour les autres, le rafraichissement automatique des vues peut être remplacé par un rappel manuel à la bibliothèque de templating sur une sous-partie d'un template ; et les saisies utilisateur peuvent être gérées de manière plus classique en définissant soi-même les EventListeners. Moins de magie, plus de contrôle, ce n'est parfois pas plus mal.

7.2. Gérer l'internationalisation côté client

Beaucoup de sites utilisent les solutions de templating côté serveur pour gérer également la traduction du site en différentes langues selon l'utilisateur. Gérer l'internationalisation en JavaScript est également possible, bien que le sujet ne soit pas beaucoup abordé dans les bibliothèques de templating. Vous pouvez tout à fait faire cohabiter dans les réponses serveur vos données avec différents labels traduits dans la langue de l'utilisateur. En revanche ce n'est peut-être pas la solution optimale. En effet, il y a généralement beaucoup plus de texte à traduire que de données à afficher au sein d'une page. Cela pourrait entraîner une surcharge importante des réponses AJAX et une perte de performance.
Dans un de mes projets, j'ai imaginé une autre solution. Mon site est une Single Page Application comme décrit en introduction, et donc tout le HTML de mes pages passe par une fonction JavaScript avant d'être injecté dans le document. J'ai ajouté un traitement supplémentaire en JavaScript juste avant l'injection : tout le HTML est analysé avec une expression régulière pour identifier un certain pattern et remplacer les labels par l'équivalent traduit. Les traductions sont chargées préalablement au format JSON lors du chargement initial du site.

```
app.injectHTML = function(container, html){
        var htmlTraduit = html.replace(/{LANG\.
([\w\.]+)\}/g, function(match, label){
            if(label in app.lang){
                return app.lang[label];
            } else {
                console.warn("Label "+label+" not
found in lang "+app.lang.name);
                return "???";
            }
        });
        $(container).html(html);
    };

    app.lang = {
        "name": "FR",
        "accueil.bienvenue": "Bonjour
{{username}} !",
        ...
    };
```

Ce qui est intéressant avec cette solution, c'est que ces labels traduits peuvent contenir d'autres marqueurs de template utilisés avec un autre outil de templating en aval (dans le cas présent, une syntaxe mustache). Cela donne pas mal de flexibilité dans le formatage des données selon la langue. La principale contrainte de cette solution est que l'on doit charger toutes les traductions au préalable. Le but est donc de trouver le meilleur moment pour envoyer les labels traduits sans que cela se ressente sur les performances.
Je ne prétends pas que cette solution soit la meilleure, et si vous avez d'autres idées n'hésitez pas à les faire partager dans les commentaires. Bien sûr, ce genre de mécanismes peut s'appliquer à d'autres choses que la gestion des langues : du chargement adaptatif selon le navigateur ou le type d'appareil par exemple…

8. Conclusion

Voilà, vous savez à présent tout du templating client. À présent, vous n'avez plus d'excuses pour écrire du JavaScript interminable et illisible quand il s'agit de générer du HTML. J'espère que cet article vous a été profitable, et je compte sur vous pour venir enrichir le sujet avec vos propres expériences.

Retrouvez l'article de Sylvain Pollet-Villard en ligne : Lien 35

Microblogging avec le MkFramework - Créez votre propre twitter-like en moins d'une heure

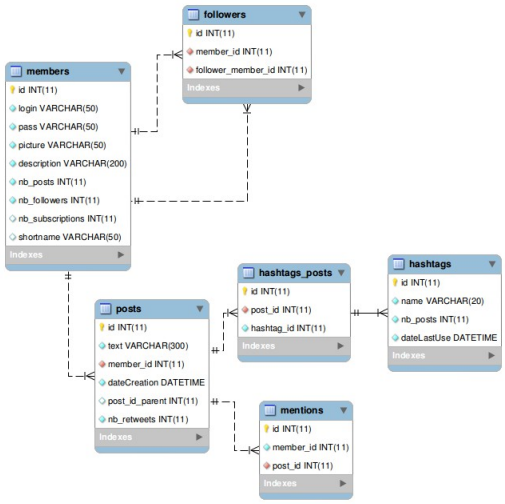
Dans le précédent tutoriel (Lien 36), vous avez découvert ce framework. Je vous propose ici d'apprécier sa simplicité et sa productivité.

1. Présentation du tutoriel

Dans ce tutoriel, nous allons apprendre à utiliser le framework pour créer une application de microblogging (comme le célèbre oiseau).

2. Le modèle de données

Voici les tables à créer pour notre tutoriel.



Ci-dessous le script SQL de création de la base de données « microblogging » :

```
CREATE TABLE members (
id int(11) NOT NULL AUTO_INCREMENT,
login varchar(50) NOT NULL,
pass varchar(50) NOT NULL,
picture varchar(50) NOT NULL,
description varchar(200) NOT NULL,
nb_posts int(11) NOT NULL,
nb_followers int(11) NOT NULL,
nb_subscriptions int(11) DEFAULT NULL,
shortname varchar(50) DEFAULT NULL,
PRIMARY KEY (id)
);

CREATE TABLE posts(
id int(11) NOT NULL auto_increment,
text varchar(300) NOT NULL,
member_id int(11) NOT NULL,
dateCreation datetime NOT NULL,
post_id_parent int(11) NULL,
nb_retweets int(11) NOT NULL,
PRIMARY KEY (id)
);

CREATE TABLE hashtags (
id int(11) NOT NULL AUTO_INCREMENT,
name varchar(20) NOT NULL,
nb_posts int(11) NOT NULL,
dateLastUse datetime NOT NULL,
PRIMARY KEY (id)
);

CREATE TABLE hashtags_posts (
id int(11) NOT NULL AUTO_INCREMENT,
post_id int(11) NOT NULL,
hashtag_id int(11) NOT NULL,
PRIMARY KEY (id)
);

CREATE TABLE mentions (
id int(11) NOT NULL AUTO_INCREMENT,
member_id int(11) NOT NULL,
post_id int(11) NOT NULL,
PRIMARY KEY (id)
);

CREATE TABLE followers (
id int(11) NOT NULL AUTO_INCREMENT,
member_id int(11) NOT NULL,
follower_member_id int(11) NOT NULL,
PRIMARY KEY (id)
);
```

3. L'application de base

3.1. Création de l'application

Rendez-vous sur l'adresse du builder : Lien 37
Renseignez « microblogging » puis validez avec le bouton « Créer ».

3.2. Présentation de l'arborescence

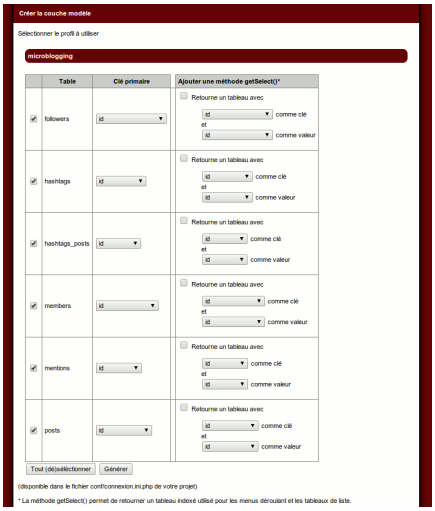
Éditez le fichier conf/connexion.ini.php.
Renseignez votre profil de connexion :

```
microblogging.dsn="mysql:dbname=microblogging;host=localhost"
microblogging.sgbd=pdo_mysql
microblogging.hostname=localhost
microblogging.database=microblogging
microblogging.username=root
microblogging.password=root
```

Connexion à une base de données « microblogging » via « pdo_mysql », avec l'utilisateur « root » et le mot de passe « root » également (à modifier en fonction des paramètres de votre base de données).

3.3. Génération de la couche modèle

Cliquez sur « générer la couche modèle ».
Sélectionnez votre connexion à la base de données (« microblogging ») :



Cochez les tables créées précédemment.
Validez avec le bouton « Générer ».
Un fichier par table va être créé dans le répertoire « model » de votre projet.

4. Les modules

4.1. Introduction

Pour cette application, nous allons créer plusieurs modules afin de gérer les différentes fonctionnalités.
Un module, si l'on compare aux autres frameworks (Zend framework, Symfony…), est composé d'un contrôleur et de ses vues.

4.2. Module public et d'authentification

4.2.1. Introduction

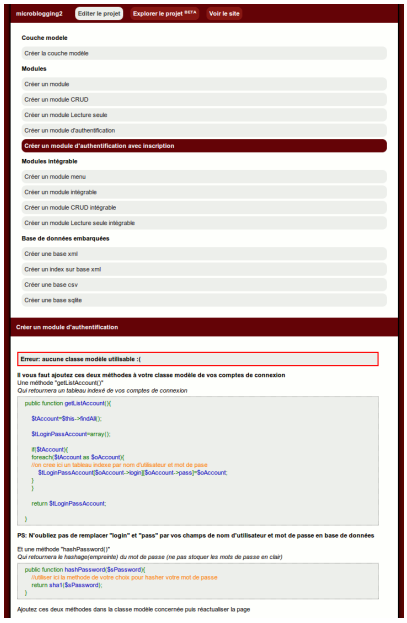
Cette application va nécessiter une authentification, c'est-à-dire un formulaire avec un nom d'utilisateur/mot de passe pour identifier le membre. Pour cela nous allons créer un module qui contiendra trois pages (aussi appelées « actions ») (login/logout/inscription).

4.2.2. Création d'un module d'authentification

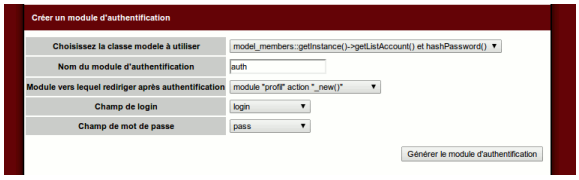
Le « builder » contient un formulaire pour créer des modules et des modules d'authentification, celui-ci créé :

- un sous-répertoire dans le répertoire « module » de votre projet ;
- un fichier contrôleur main.php ;
- un sous-répertoire « view » contenant les fichiers de vues nécessaires.

Cliquez sur « Créer un module d'authentification avec inscription ».



Vous allez voir s'afficher une page d'erreur vous indiquant que vous ne possédez pas de classe modèle éligible pour créer ce module, et vous demandant d'ajouter deux méthodes à la classe modèle contenant vos comptes de connexion.
Une méthode getListAccount() qui retournera l'ensemble des comptes dans un tableau indexé ainsi qu'une méthode hashPassword() qui hachera votre mot de passe.
Ajoutez ces deux méthodes à votre classe modèle model_members.php puis retournez sur ce menu : il n'y a plus d'erreur.



Saisissez dans le champ « module » : « auth », sélectionnez vers quel module diriger une fois authentifié, et indiquez via les menus déroulants les champs de login et de mot de passe.
Cliquez ensuite sur « Générer ».
Le builder va créer :

- un répertoire « module/auth » ;
- un fichier module/auth/main.php (contenant le contrôleur du module) ;
- un sous répertoire « module/auth/view » ;
- et deux fichiers de vues login.php et inscription.php dans le répertoire « module/auth/view ».

Vous allez également lire que vous devez modifier le fichier de paramétrage pour activer l'authentification sur l'ensemble du site en passant la variable « enabled » à 1 dans la section [auth] du fichier conf/site.ini.php.

4.2.3. Test du module d'authentification

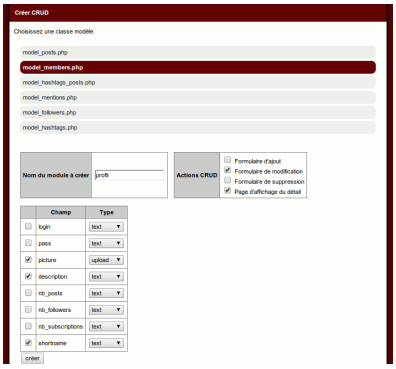
Profitez-en pour tester ce départ : cliquez sur le lien inscription, saisissez un nom d'utilisateur et un mot de passe et enregistrez-le.
Par exemple « login » comme nom d'utilisateur et « pass »

comme mot de passe.
Vous pouvez désormais vous authentifier avec votre nouveau compte.

4.3. Module profil

4.3.1. Introduction

Actuellement, lorsque l'on se connecte, on arrive sur une page vide, on va donc créer un module de profil qui nous permettra de voir et modifier notre profil (avatar, informations…). Pour cela, nous allons utiliser le builder.
Cliquez sur « Éditer le projet » en face de votre projet.
Cliquez ensuite sur « Créer un module CRUD », puis sur la classe modèle « model_members.php ».



Vous avez un formulaire de création de CRUD, modifiez « members » (nom de la table) par « profil ».
Décochez les cases « Formulaire d'ajout », « Formulaire de suppression » (nous n'en avons pas besoin).
Décochez ensuite les champs « nb_posts », « nb_followers », « nb_subscriptions », « login » et « pass ».
Pour le champ « picture », dans le menu déroulant, sélectionnez « upload ».
Puis cliquez sur « Créer ».

Le builder va générer un répertoire « profil » dans le répertoire « module », un fichier contrôleur main.php ainsi qu'un sous-répertoire « view » contenant quatre fichiers de vue : list.php, show.php, edit.php et delete.php.

4.3.2. Un peu de ménage

Nous allons supprimer l'action et la vue inutile de ce module, éditez le fichier module/profil/main.php et supprimez la méthode _list().
idem pour le fichier module/profil/view/list.php.

4.3.3. Visualisation de son profil

Nous souhaitons par défaut que ce module affiche notre profil en lecture, pour cela on va modifier notre module CRUD.
Éditez le fichier module/profil/main.php.
On remplace notre action _index() (page par défaut) :

```
public function _index(){
    //on considere que la page par defaut est la page de listage
    $this->_list();
}
```

Par :

```
public function _index(){
    //on considere que la page par defaut est la page d'affichage de profil
    $this->_show();
}
```

On édite ensuite la méthode _show(), où l'on force l'id du membre affiché par celui de notre membre connecté.
L'appel à _root::getAuth()>getAccount() nous retourne l'objet du membre connecté.

```
public function _show(){
    //recuperation de l'id de notre membre connecte
    $member_id=_root::getAuth()->getAccount()->id;
    $oMembers=model_members::getInstance()->findById( $member_id );
    $oView=new _view('profil::show');
    $oView->oMembers=$oMembers;
    $this->oLayout>add('main',$oView);
}
```

On modifie ensuite la vue pour y ajouter un bouton d'édition du profil.
Fichier module/profil/view/show.php.

```
<table class="tb_show">
    <tr>
        <th>shortname</th>
    <td><?php echo $this->oMembers->shortname ?></td>
    </tr>
    <tr>
        <th>picture</th>
    <td><?php echo $this->oMembers->picture ?></td>
    </tr>
    <tr>
        <th>description</th>
    <td><?php echo $this->oMembers->description ?></td>
    </tr>
</table>
<p style="textalign:right"> <a href="<?php echo $this>getLink('profil::edit')?>">Éditer</a></p>
```

4.3.4. L'édition de son profil

Nous allons ensuite, comme pour la méthode d'affichage, forcer le membre édité par celui connecté. Éditez la méthode _edit() toujours dans module/profil/main.php.
Remplacez :

```
public function _edit(){
    $tMessage=$this->save();
    $oMembers=model_members::getInstance()->findById( _root::getParam('id') );
    $oView=new _view('profil::edit');
    $oView->oMembers=$oMembers;
    $oView->tId=model_members::getInstance()->getIdTab();
    $oPluginXsrf=new plugin_xsrf();
    $oView->token=$oPluginXsrf->getToken();
    $oView->tMessage=$tMessage;
    $this->oLayout->add('main',$oView);
```

```
}
```

Par :

```
public function _edit(){
    $tMessage=$this->save();
    //recuperation de l'id de notre membre
connecte
    $member_id=_root::getAuth()->getAccount()-
>id;
    $oMembers=model_members::getInstance()-
>findById( $member_id );
    $oView=new _view('profil::edit');
    $oView->oMembers=$oMembers;
    $oView->tId=model_members::getInstance()-
>getIdTab();
    $oPluginXsrf=new plugin_xsrf();
    $oView->token=$oPluginXsrf->getToken();
    $oView->tMessage=$tMessage;
    $this->oLayout->add('main',$oView);
}
```

Il faut également forcer la redirection post-modification vers la page d'affichage du profil.
Et forcer à toujours modifier (supprimer l'ajout de membre).
Remplacez dans la méthode save() :

```
$iId=_root::getParam('id',null);
        if($iId==null){
            $oMembers=new row_members;
        }else{

$oMembers=model_members::getInstance()->findById(
_root::getParam('id',null) );
        }
```

Par :

```
//recuperation de l'id de notre membre connecte
$member_id=_root::getAuth()->getAccount()->id;
$oMembers=model_members::getInstance()->findById(
$member_id );
```

Et :

```
//une fois enregistre on redirige (vers la page
liste)
_root::redirect('profil::list');
```

Par :

```
//une fois enregistre on redirige (vers la page
d'affichage)
_root::redirect('profil::show');
```

Ici vous pouvez éditer votre profil, choisir une image, définir un shortname et une description et valider.
Mais lorsque le profil s'affiche, on voit le chemin de l'image et non l'image, modifions un peu la vue.
Éditez le fichier module/profil/view/show.php.

```
<div style="float:right;border:2px solid
gray"><img style="width:100px" src="<?php echo
$this->oMembers->picture ?>"/></div>
<table class="tb_show">
    <tr>
        <th>shortname</th>
        <td><?php echo $this->oMembers->shortname
?></td>
    </tr>
    <tr>
        <th>description</th>
        <td><?php echo $this->oMembers-
>description ?></td>
    </tr>
</table>
<div style="clear:both"></div>
<p style="textalign:right"> <a href="<?php echo
$this->getLink('profil::edit')?>">Éditer</a></p>
```

4.3.5. Comment forcer ce module lorsque l'on se connecte

Éditez le fichier module/default/index.php et remplacez dans la méthode _index() :

```
public function _index(){
    $oView=new _view('default::index');
    $this->oLayout->add('main',$oView);
}
```

Par :

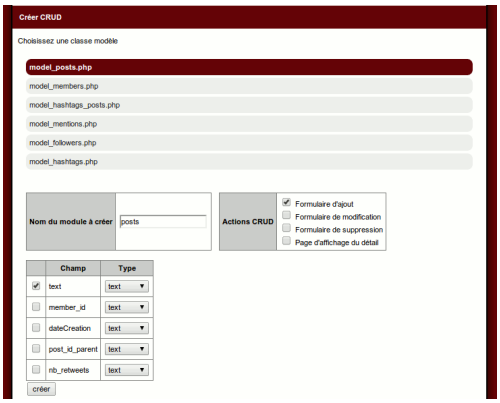
```
public function _index(){
    _root::redirect('profil::index');
}
```

Nous allons rediriger par défaut sur la page de profil.

4.4. Module posts

4.4.1. Introduction

Passons aux posts : la partie principale de l'application, pour faire simple nous allons d'abord créer un module CRUD que nous modifierons pour répondre à notre besoin.
Rendez-vous sur le builder et cliquez sur « Créer un module CRUD », sélectionnez le modèle model_post.php.



Désélectionnez les cases « Formulaire de modification », « Formulaire de suppression » et « Page affichage de détail ».
Désélectionnez tous les champs sauf « ext ».

4.4.2. Amélioration du modèle model_posts

Ajoutez la méthode suivante dans le fichier model_post.php.
Dans la classe model_posts :

```
public function
findAllOwnerAndFollowed($members_id){
    return $this->findMany('SELECT posts.* FROM
'.$this->sTable.' LEFT OUTER JOIN followers ON
    followers.member_id=posts.member_id WHERE
posts.member_id=? OR
followers.follower_member_id=? ORDER BY posts.id
DESC',$members_id,$members_id);
}
```

Dans la classe row_posts on va ajouter une méthode pour récupérer son auteur.
Ajoutez la méthode suivante dans la classe row_posts :

```
public function findMember(){
    return model_members::getInstance()-
>findById($this->member_id);
}
```

Modifiez également la méthode d'enregistrement pour ajouter la date à l'insertion en base. Dans la méthode save() de la classe row_posts juste avant l'appel à la méthode save() de la classe parente.
Ajoutez :

```
$this->dateCreation=date('Y-m-d H:i:s');
```

Ce qui donne :

```
public function save(){
    if(!$this->isValid()){
        return false;
    }
    $this->dateCreation=date('Y-m-d H:i:s');
    parent::save();
    return true;
}
```

4.4.3. Visualisation de son fil (avec post des abonnements + les siens)

Il faut utiliser la méthode précédemment ajoutée pour afficher uniquement les posts de l'utilisateur.
Modifiez le fichier main.php du module posts pour remplacer dans la méthode _list() :

```
$tPosts=model_posts::getInstance()->findAll();
```

Par :

```
$tPosts=model_posts::getInstance()-
>findAllOwnerAndFollowed(_root::getAuth()-
>getAccount()->id);
```

Modifions la vue liste.
Éditez le fichier module/posts/view/list.php :

```
<h1>Tweets</h1>
<table class="tb_posts">
    <?php if($this->tPosts):?>
        <?php foreach($this->tPosts as $oPosts):?
>
            <?php
            $oMember=$oPosts->findMember();
            $oDate=new plugin_datetime($oPosts-
>dateCreation);
            $sDate=$oDate->toString('d/m/Y
&\a\g\r\a\v\e; H\hi');
            $sText=$oPosts->text;
            ?>
            <tr <?php echo
plugin_tpl::alternate(array('','class="alt"'))?>>
            <td style="width:22px"><img
style="width:20px" src="<?php echo $oMember-
>picture?>"/></td>
            <td>
            <div
style="float:right;color:gray"><?php echo $sDate?
></div>
            <strong><?php echo $oMember-
>shortname?></strong> <a style="color:#444"
            href="<?php echo
_root::getLink('profil::showother',array('name'=>
$oMember->login))?>">@<?php echo $oMember->login?
></a><br/><?php echo $sText ?></td>
            </tr>
        <?php endforeach;?>
        <?php else:?>
        Aucun posts
    <?php endif;?>
</table>
<p><a href="<?php echo $this-
>getLink('posts::new') ?>">Nouveau
message</a></p>
```

4.4.4. Poster

Si vous utilisez le lien « new », vous allez ajouter un enregistrement dans la table posts, mais sans indiquer son auteur, on va forcer l'auteur à l'enregistrement.
Modifions la méthode save() dans le fichier main.php du module « posts ».
Vous allez ajouter une ligne pour forcer le member_id avant l'enregistrement :

```
//on force l'id du membre
$oPosts->member_id=_root::getAuth()-
>getAccount()->id;
```

Elle ressemblera à ceci :

```
public function save(){
    if(!_root::getRequest()->isPost() ){ //si ce
n'est pas une requete POST on ne soumet pas
        return null;
    }
    $oPluginXsrf=new plugin_xsrf();
    if(!$oPluginXsrf-
>checkToken( _root::getParam('token') ) ){ //on
verifie que le token est valide
        return array('token'=>$oPluginXsrf-
>getMessage() );
    }
    $iId=_root::getParam('id',null);
    if($iId==null){
        $oPosts=new row_posts;
    }else{
        $oPosts=model_posts::getInstance()-
>findById( _root::getParam('id',null) );
    }
    $tId=model_posts::getInstance()->getIdTab();
    $tColumn=model_posts::getInstance()-
>getListColumn();
    foreach($tColumn as $sColumn){
        if(isset($_FILES[$sColumn]) and
$_FILES[$sColumn]['size'] > 0){

$sNewFileName=_root::getConfigVar('path.upload').
$sColumn.'_'.date('Ymdhis');
```

```
            $oPluginUpload=new
plugin_upload($_FILES[$sColumn]);
            $oPluginUpload-
>saveAs($sNewFileName);
            $oPosts->$sColumn=$oPluginUpload-
>getPath();
            continue;
        }else  if( _root::getParam($sColumn,null)
=== null ){
            continue;
        }else if( in_array($sColumn,$tId)){
            continue;
        }
        $oPosts-
>$sColumn=_root::getParam($sColumn,null) ;
    }
    //on force l'id du membre
    $oPosts->member_id=_root::getAuth()-
>getAccount()->id;
    if($oPosts->save()){
        //une fois enregistre on redirige (vers
la page liste)
        _root::redirect('posts::list');
    }else{
        return $oPosts->getListError();
    }
}
```

4.5. Module hashtags

4.5.1. Récupération du hashtag par son nom

Ajoutez une méthode dans model_hashtags.php.

```
public function findByName($uId){
    return $this->findOne('SELECT * FROM '.
$this>sTable.' WHERE name=?',$uId );
}
```

4.5.2. Modification du modèle hashtags pour récupérer les hashtags d'un post

Éditez le fichier model_hashtags.php.
Et ajoutons une méthode pour récupérer les hashtags :

```
public function findAllHashtags($sText){
    $sText.=' ';
    preg_match_all('/#([0-9a-zA-Z]*) /',$sText,
$tHashtag);
    if(isset($tHashtag[1])){
        return $tHashtag[1];
    }
    return null;
}
```

4.5.3. Ajout d'une méthode pour prendre en compte les hashtags

Toujours le même fichier modèle model_hashtags.
Nous allons ajouter une méthode qui recevra un tableau de hashtags et créera les liens SQL entre ceux-ci et notre post :

```
public function addForPost($sHashtag,$post_id){
    //recherche du hashtag par son nom
    $oHashtag=$this->findByName($sHashtag);
    if($oHashtag){
        //si on trouve le hashtag en base, on
incremente sa notoriete
        $oHashtag->nb_posts=$oHashtag-
>nb_posts+1;
    }else{
        //si on ne le trouve pas on créé ce
hashtag
        $oHashtag=new row_hashtags;
        $oHashtag->name=$sHashtag;
        $oHashtag->nb_posts=1;
    }
    //dans les deux cas on indique au hashtag sa
dernière utilisation
    $oHashtag->dateLastUse=date('Ymd H:i:s');
    $oHashtag->save();
    //enfin on sauvegarde le lien post/hashtag
    $oHashtagPost=new row_hashtags_posts;
    $oHashtagPost->post_id=$post_id;
    $oHashtagPost->hashtag_id=$oHashtag->id;
    $oHashtagPost->save();
}
```

4.5.4. À l'enregistrement d'un post, enregistrer les hashtags

On va prendre en compte ici les méthodes ajoutées précédemment.
Dans la classe model_post, profitons de la méthode save() pour prendre en compte les hashtags.
Éditez la classe row_posts :

```
public function save(){
    if(!$this->isValid()){
        return false;
    }
    $this>dateCreation=date('Ymd H:i:s');
    parent::save();
    //on en profite pour enregistrer les tags et
les mentions
    //les hashtags
    $tHashtags=model_hashtags::getInstance()-
>findAllHashtags($this->text);
    if($tHashtags){
        foreach($tHashtags as $sHashtag){
            $sHashtag=trim($sHashtag);
            model_hashtags::getInstance()-
>addForPost($sHashtag,$this->id);
        }
    }
    return true;
}
```

4.5.5. Visualisation des hastags les plus populaires

Créons un module pour voir les hashtags.
Passez par le builder pour créer un module « hashtags » avec deux actions « list » et « show ».
Éditez ensuite le fichier module/hashtags/main.php pour afficher le menu.
Modifiez la méthode before() :

```
public function before(){
    $this->oLayout=new _layout('template1');
    $this->oLayout-
>addModule('menu','menu::list');
}
```

Modifiez également la méthode _list() pour récupérer les hashtags et les ajouter à la vue :

```
public function _list(){
    //recuperons la liste des hashtags

$tHastags=model_hashtags::getInstance()>findAll()
;
    $oView=new _view('hashtags::list');
    //assignons le tableau d'hashtag a la vue
    $oView->tHastags=$tHastags;
    $this->oLayout->add('main',$oView);
}
```

Modifiez également la vue list module/hashtags/view/list.php.
Pour lister les hashtags :

```
<h1>Hashtags</h1>
<table class="tb_posts">
        <?php if($this->tHastags):?>
        <?php foreach($this->tHastags as
$oHastags):?>
                <tr <?php echo
plugin_tpl::alternate(array('','class="alt"'))?>>
                 <td><strong><a href="<?php echo
_root::getLink('hashtags::show',array('name'=>$oH
astags->name))?>">#<?php echo $oHastags->name ?
></a></strong>
                                <br /><strong><?php
echo $oHastags->nb_posts ?></strong> TWEETS <br/>
                                    <i>dernier tweet
le <?php echo $oHastags->dateLastUse ?></i>
                            </td>
                </tr>
        <?php endforeach;?>
        <?php endif;?>
</table>
```

4.5.6. Affichage des hashtags et surtout des posts associés

Occupons-nous de la page pour afficher les hashtags et ses posts associés.
Au départ on ajoute une méthode dans le modèle posts pour lister les posts associés à un hashtag.
Ouvrez le fichier model_posts.php :

```
public function findByHashtag($uId){
    return $this->findMany('SELECT posts.* FROM
'.$this->sTable.',hashtags_posts WHERE
posts.id=hashtags_posts.post_id AND
hashtag_id=?',$uId );
}
```

Ensuite ajoutez une méthode pour retourner les posts liés à un hashtag.
Dans le module module/posts/main.php :

```
public function listByHashtag($hashtag_id){
        $tPosts=model_posts::getInstance()-
>findByHashtag($hashtag_id);

        $oView=new _view('posts::list');
        $oView->tPosts=$tPosts;

    return $oView;
}
```

Retrouvez la suite de l'article de Michael Bertocchi en ligne : Lien 38

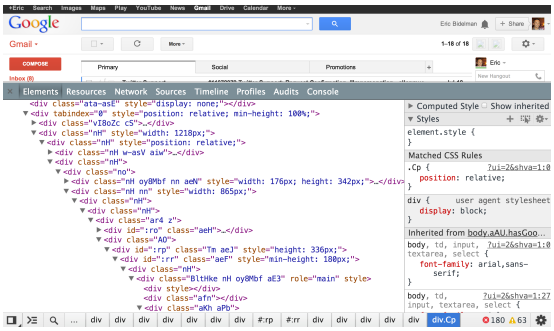
Les éléments personnalisés - Créez de nouvelles balises HTML

Attention : cet article présente une API qui n'est pas encore finalisée et susceptible d'évoluer. Soyez prudents lorsque vous utilisez des API expérimentales dans vos projets.

Cet article est la traduction de Custom Elements : defining new elements in HTML publié sur le site HTML5 Rocks : Lien 39.

1. Introduction

Le Web manque cruellement d'expressivité. Pour comprendre ce que j'entends par-là, prenez une application Web « moderne » comme GMail :



Applications Web modernes : créées avec des soupes de balises

Il n'y a rien de moderne avec une telle soupe de balises . Pourtant… c'est actuellement comme ça que nous construisons ce type d'applications. C'est malheureux. Ne devrions-nous pas attendre mieux de nos plateformes ?

1.1. Rendre possible un balisage agréable

HTML nous offre un excellent outil pour structurer un document, mais son vocabulaire est limité aux éléments définis dans les spécifications HTML : Lien 40.
À quoi ressemblerait GMail si son balisage n'était pas horrible ? Comment pourrait-on le rendre plus beau :

```
<hangout-module>
  <hangout-chat from="Paul, Addy">
    <hangout-discussion>
      <hangout-message from="Paul"
profile="profile.png"
          profile="11807591949662637579l"
datetime="2013-07-17T12:02">
        <p>Feelin' this Web Components thing.</p>
        <p>Heard of it?</p>
      </hangout-message>
    </hangout-discussion>
  </hangout-chat>
  <hangout-chat>...</hangout-chat>
</hangout-module>
```

C'est beaucoup plus élégant ! Surtout, cette application a du sens. Elle est **éloquente**, **facile à comprendre** et surtout, **maintenable**. N'importe quel utilisateur futur saura exactement ce qu'elle fait juste en regardant son ossature.

Allez les éléments personnalisés, aidez-nous ! Vous êtes notre seul espoir.

2. Pour démarrer

L'API Custom Elements (Lien 41) **permet aux développeurs Web de définir de nouveaux types d'éléments HTML**. La spécification est l'une des nouvelles API majeures apparues sous la bannière des composants Web (Web Components) (Lien 42), mais elle est surtout probablement la plus importante d'entre elles. D'ailleurs, les composants Web ne peuvent pas exister sans les fonctionnalités apportées par les éléments personnalisés :

- définir de nouveaux éléments HTML/DOM ;
- créer des éléments qui en étendent d'autres ;
- regrouper dans une balise unique des fonctionnalités liées ;
- étendre l'API d'éléments DOM existants.

2.1. Enregistrer de nouveaux éléments

Les éléments personnalisés sont créés avec document.register() :

```
var XFoo = document.register('x-foo');
document.body.appendChild(new XFoo());
```

Le premier paramètre de document.register() correspond au nom de la balise créée. Ce nom **doit obligatoirement contenir un tiret (-)**. Par exemple, <x-tags>, <my-element> et <my-awesome-app> sont des noms valides, alors que <tabs> et <foo_bar> ne le sont pas. Cette restriction permet au parseur de différencier les éléments personnalisés des éléments natifs, mais permet surtout d'assurer une compatibilité ascendante lorsque de nouvelles balises sont ajoutées au HTML.
Le second paramètre (optionnel) est un objet décrivant le prototype de l'élément. C'est ici qu'il faudra définir les fonctionnalités (par exemple ses propriétés et méthodes publiques) de vos éléments. Nous y reviendrons chapitre 4.
Par défaut, les nouveaux éléments héritent de l'objet HTMLElement. Ainsi, l'exemple précédent est équivalent à :

```
var XFoo = document.register('x-foo', {
  prototype: Object.create(HTMLElement.prototype)
});
```

Un appel à document.register('x-foo') permet au navigateur de reconnaître le nouvel élément et retourne un constructeur que vous pouvez utiliser pour créer des instances de <x-foo>. Vous pouvez également utiliser d'autres techniques pour instancier des éléments (voir chapitre 3) si vous ne souhaitez pas utiliser le constructeur.

Astuce
Si vous ne souhaitez pas que le constructeur soit dans le contexte global, placez-le dans un espace de nommage : var myapp = {}; myapp.XFoo = document.register('x-foo');

2.2. Étendre des éléments natifs

Imaginons que vous ne soyez pas particulièrement satisfait de la balise <button>. En fait, vous aimeriez l'enrichir pour la faire devenir un « mega bouton ». Pour étendre l'élément <button>, il suffit de créer un élément personnalisé qui hérite du prototype de HTMLButtonElement :

```
var MegaButton = document.register('mega-button',
{
  prototype:
Object.create(HTMLButtonElement.prototype)
});
```

À savoir
Pour créer un **élément A** qui étende un **élément B**, l'élément A doit hériter du prototype de l'élément B.

Cette catégorie d'éléments personnalisés s'appelle « élément personnalisé par extension de type » (type extension custom elements). Ils héritent d'une version spécifique de HTMLElement comme si on disait « l'élément X est un Y ».
Exemple :

```
<button is="mega-button">
```

2.3. Comment sont gérés les éléments

Ne vous êtes-vous jamais demandé pourquoi les parseurs HTML des navigateurs ne sont pas perturbés lorsqu'ils rencontrent des éléments non standards ? Par exemple, vous pouvez mettre une balise <randomtag> dans votre code et le navigateur ne s'en portera pas plus mal. En fait, la spécification HTML (Lien 43) prévoit ce cas :

The HTMLUnknownElement interface must be used for HTML elements that are not defined by this specification.

Ce qui peut se traduire par :

L'interface HTMLUnknownElement doit être utilisée pour les éléments HTML non définis par cette spécification.

Désolé <randomtag>, tu es inconnu des standards et hériteras de HTMLUnknownElement.
Mais cela ne s'applique pas aux éléments personnalisés. **Les éléments personnalisés ayant un nom valide héritent de** HTMLElement. Vous pouvez vérifier cela en activant votre console JavaScript et en exécutant le code ci-après, les exemples renvoient true :

```
// "tabs" n'est pas un nom d'élément personnalisé
valide
document.createElement('tabs').__proto__ ===
HTMLUnknownElement.prototype

// "x-tabs" est un nom d'élément personnalisé
valide
document.createElement('x-tabs').__proto__ ==
HTMLElement.prototype
```

Note
<x-tabs> sera malgré tout de type HTMLUnknownElement pour les navigateurs ne supportant pas document.register().

2.3.1. Éléments non résolus

Comme les nouveaux éléments sont signifiés au navigateur par script via document.register(), **il se peut qu'ils soient utilisés avant que leur définition ne soit interprétée**. Par exemple, vous pouvez utiliser une balise <x-tab> dans le HTML, mais n'appeler document.register('x-tabs') que plus tard.
Avant que les éléments ne soient rattachés à leur définition, ils sont dits « non résolus » (unresolved elements). Il s'agit des éléments ayant un nom valide, mais n'ayant pas encore été enregistrés.
Voici un tableau qui vous aidera à y voir plus clair.

Nom	Hérite de	Exemples
Élément non résolu	HTMLElement	<x-tabs>, <my-element>, <my-awesome-app>
Élément inconnu	HTMLUnknownElement	<tabs>, <foo_bar>

On peut assimiler les éléments non résolus à des oublis. Ils sont potentiellement éligibles pour être validés par le navigateur ultérieurement. Le navigateur estime qu'ils ont toutes les propriétés requises pour être de nouveaux éléments et attend leur définition pour les valider.

3. Instancier les éléments

La méthode habituelle pour créer des éléments s'applique aussi aux éléments personnalisés. Comme tout élément standard, ils peuvent être présents dans le HTML ou créés dans le DOM à l'aide de JavaScript.

3.1. Instancier de nouvelles balises

Déclarez-les :

```
<x-foo></x-foo><x-foo></x-foo>
```

Créez un objet DOM en JavaScript :

```
var xFoo = document.createElement('x-foo');
xFoo.addEventListener('click', function(e) {
  alert('Thanks!');
});
```

Utilisez l'opérateur new :

```
var xFoo = new XFoo();
document.body.appendChild(xFoo);
```

3.2. Instancier des éléments par extension de type

Instancier des extensions d'éléments existants se fait de façon similaire.
Déclarez-les :

```
<!-- <button> "est un" mega bouton -->
<button is="mega-button">
```

Créez un objet DOM en JavaScript :

```
var megaButton = document.createElement('button',
'mega-button');
// megaButton instanceof MegaButton === true
```

Comme vous pouvez le voir, il existe maintenant une version enrichie de document.createElement() qui prend en second paramètre la valeur de l'attribut is="".
Utilisez l'opérateur new :

```
var megaButton = new MegaButton();
document.body.appendChild(megaButton);
```

Jusqu'à présent, nous n'avons utilisé que document.register() pour ajouter de nouveaux éléments, mais cela ne va pas très loin… alors, ajoutons des propriétés et des méthodes.

4. Ajouter des propriétés et des méthodes JavaScript

La puissance des éléments créés par l'utilisateur réside dans le fait de pouvoir leur attribuer des fonctionnalités particulières en leur attribuant des propriétés et des méthodes spécifiques dans leur définition. Cela revient à fournir une API publique liée à vos éléments.
Voici un exemple complet :

```
var XFooProto =
Object.create(HTMLElement.prototype);

// 1. Donner à x-foo une méthode foo().
XFooProto.foo = function() {
  alert('foo() called');
};

// 2. Définir une propriété en lecture seule
"bar".
Object.defineProperty(XFooProto, "bar", {value:
5});

// 3. Enregistrer la définition de x-foo.
var XFoo = document.register('x-foo', {prototype:
XFooProto});

// 4. Instancier un objet x-foo.
var xfoo = document.createElement('x-foo');

// 5. L'ajouter à la page.
document.body.appendChild(xfoo);
```

Bien sûr, il existe une multitude de façons de créer un prototype. Si la version précédente n'est pas votre préférée, en voici une plus condensée :

```
var XFoo = document.register('x-foo', {
  prototype: Object.create(HTMLElement.prototype,
{
    bar: {
      get: function() { return 5; }
    },
    foo: {
      value: function() {
        alert('foo() called');
      }
    }
  })
});
```

Le premier exemple utilise la méthode ES5 Object.defineProperty (Lien 44), la seconde utilise les méthodes get/set (Lien 45).

4.1. Les fonctions de rappel au cours du cycle de vie

Les éléments permettent de définir des méthodes spéciales liées aux moments importants de leur mise en œuvre. Ces méthodes sont appelées de façon avisée « fonctions de rappel au cours du cycle de vie » (lifecycle callbacks). Chacune a un nom et une utilité spécifique.

Nom de la fonction de rappel	Appelée quand…
createdCallback	une instance de l'élément est créée
enteredDocumentCallback	une instance est intégrée dans le document
leftDocumentCallback	une instance est retirée du document
attributeChangedCallback(attrName, oldVal, newVal)	un attribut a été ajouté, supprimé ou modifié

Dans l'exemple suivant, nous valorisons les fonctions de rappel createdCallback() et enteredDocumentCallback() sur la balise <x-foo> :

```
var proto = Object.create(HTMLElement.prototype);

proto.createdCallback = function() {...};
proto.enteredDocumentCallback = function() {...};

var XFoo = document.register('x-foo', {prototype:
proto});
```

Chacune de ces fonctions de rappel est optionnelle, ne les définissez que si cela est utile. Par exemple, imaginons un élément particulièrement complexe qui ouvre une connexion à IndexedDB dans createdCallback(). Dans ce cas, n'oubliez pas, s'il doit être retiré du DOM, de faire le ménage nécessaire dans leftDocumentCallback().

Note
Vous ne devriez pas utiliser ces méthodes si l'utilisateur ferme un onglet par exemple, voyez-les plutôt comme des possibilités d'optimisation.

Un autre cas classique d'utilisation est de définir des gestionnaires d'événements par défaut :

```
proto.createdCallback = function() {
  this.addEventListener('click', function(e) {
    alert('Thanks!');
  });
};
```

Personne n'utilisera vos éléments s'ils sont bancals, les fonctions de rappel du cycle de vie vous aideront à être un bon citoyen !

5. Ajouter du balisage

Nous avons maintenant créé un élément <x-foo> et lui avons fourni une API, mais il est encore vide ! Peut-être pourrions-nous lui donner un peu de HTML à afficher ?
Les fonctions de rappel du cycle de vie sont très utiles pour ça. En particulier, nous pouvons utiliser createdCallback() pour doter un élément de code HTML par défaut à afficher :

```
var XFooProto =
Object.create(HTMLElement.prototype);

XFooProto.createdCallback = function() {
  this.innerHTML = "<b>Je suis un x-foo-with-
markup !</b>";
};

var XFoo = document.register('x-foo-with-markup',
{prototype: XFooProto});
```

Si nous instancions cette balise et que nous l'inspectons dans la console (clic droit puis « Inspecter l'élément »), nous devrions voir :

```
&#9660;<x-foo-with-markup>
   <b>Je suis un x-foo-with-markup !</b>
 </x-foo-with-markup>
```

5.1. Encapsuler le contenu dans le DOM fantôme

En soi, le DOM fantôme (Shadow DOM : Lien 46) est un outil puissant pour encapsuler du contenu. Utilisé en conjonction avec les éléments personnalisés et la magie prend forme !
Le DOM fantôme permet aux éléments personnalisés :

- de cacher leurs entrailles, permettant ainsi de masquer le cambouis de l'implémentation aux utilisateurs ;
- d'encapsuler les styles… et gratuitement : Lien 47.

Créer un élément dans le DOM fantôme lui permet de s'afficher selon un balisage de base. La différence se faisant dans createdCallback() :

```
var XFooProto =
Object.create(HTMLElement.prototype);

XFooProto.createdCallback = function() {
  // 1. Attacher une racine fantôme à l'élément.
  var shadow = this.createShadowRoot();

  // 2. Le remplir avec le bon balisage.
  shadow.innerHTML = "<b>Je suis dans le DOM
fantôme de l'élément !</b>";
};

var XFoo = document.register('x-foo-shadowdom',
{prototype: XFooProto});
```

Au lieu d'affecter le .innerHTML de l'élément, j'ai créé une racine fantôme (Shadow Root) pour <x-foo-shadowdom> et y ai ajouté le balisage. En activant l'option « Show Shadow DOM » dans les outils de développeur de Chrome (**NdT** : seul navigateur supportant cette API au moment de l'écriture de l'article), vous verrez un #document-fragment que vous pouvez déplier :

```
&#9660;<x-foo-shadowdom>
   &#9660;#document-fragment
     <b>Je suis dans le DOM fantôme de
l'élément !</b>
 </x-foo-shadowdom>
```

C'est cela la racine fantôme !

5.2. Créer des éléments à partir d'un gabarit

Les gabarits HTML (HTML Templates : Lien 48) sont une autre nouvelle API qui convient parfaitement au monde des éléments personnalisés.
Pour ceux qui ne seraient pas familiers avec la balise <template> (Lien 49), celle-ci permet de déclarer des fragments de DOM qui sont parsés, inactifs au chargement de la page et destinés à être instanciés ultérieurement. Elles sont donc l'emplacement idéal pour déclarer la structure d'un élément personnalisé.
Dans cet exemple, nous enregistrons un élément créé à partir d'un <template> dans le DOM fantôme :

```
<template id="sdtemplate">
  <style>
    p { color: orange; }
  </style>
  <p>Je suis dans le DOM fantôme. Mon balisage
provient d'un <template>.</p>
</template>

<script>
var proto = Object.create(HTMLElement.prototype,
{
  createdCallback: {
    value: function() {
      var t =
document.querySelector('#sdtemplate');

this.createShadowRoot().appendChild(t.content.clo
neNode(true));
    }
  }
});
document.register('x-foo-from-template',
{prototype: proto});
</script>
```

Cette portion de code contient beaucoup de choses. Essayons de détailler tout ce qu'il s'y passe.

- Nous enregistrons un nouvel élément HTML, <x-foo-from-template>.
- Le contenu DOM de cet élément est créé à partir d'un <template>.
- La partie interne est masquée à l'aide du DOM fantôme.

- Le DOM fantôme encapsule le style de l'élément (la règle CSS p {color: orange;} ne rend pas tous les paragraphes orange).

C'est beau !

6. Donner du style aux éléments personnalisés

Comme pour toute balise HTML, les utilisateurs vont pouvoir appliquer des styles CSS à vos éléments :

```
app-panel {
    display: flex;
}
[is="x-item"] {
    transition: opacity 400ms ease-in-out;
    opacity: 0.3;
    flex: 1;
    text-align: center;
    border-radius: 50%;
}
[is="x-item"]:hover {
    opacity: 1.0;
    background: rgb(255, 0, 255);
    color: white;
}
app-panel > [is="x-item"] {
    padding: 5px;
    list-style: none;
    margin: 0 7px;
}
```

```
<app-panel>
  <li is="x-item">Do</li>
  <li is="x-item">Re</li>
  <li is="x-item">Mi</li>
</app-panel>
```

6.1. Style des éléments du DOM fantôme

Le terrier du lapin devient beaucoup plus profond lorsque le DOM fantôme entre en jeu et les éléments personnalisés qui l'utilisent profitent de ses larges avantages.
Le DOM fantôme est parfaitement cloisonné, y compris ses déclarations de style. Les styles CSS définis à partir de la racine fantôme ne peuvent pas s'appliquer sur le reste de la page, de même, les styles de la page n'ont pas d'effet sur le DOM fantôme. **Dans le cas des éléments personnalisés, l'élément lui-même est la racine.** L'encapsulation des styles permet aussi à un élément personnalisé de définir ses propres styles par défaut.
La mise en forme du DOM fantôme est un immense sujet ! Si vous souhaitez en savoir plus à ce propos, je vous recommande certains de mes articles :

- A Guide to Styling Elements dans la documentation de Polymer : Lien 50 ;
- Shadow DOM 201 : CSS & Styling sur HTML5 Rocks : Lien 51.

6.2. Éviter le FOUC avec :unresolved

Pour limiter le FOUC, la spécification des éléments personnalisés introduit le nouveau pseudoélément :unresolved. Vous pouvez l'utiliser pour cibler les éléments non résolus jusqu'à ce que le navigateur n'invoque votre fonction createdCallback() (voir Les fonctions de rappel au cours du cycle de vie). Une fois cette fonction appelée, l'élément n'est plus « non résolu », le processus de mise à jour est achevé et l'élément est transformé conformément à sa déclaration.

Le pseudoélément :unresolved est supporté nativement dans Chrome 29.

Exemple : faire apparaître progressivement les balises <x-foo>

```
x-foo {
    opacity: 1;
    transition: opacity 300ms;
}
x-foo:unresolved {
    opacity: 0;
}
```

Gardez bien à l'esprit que le pseudoélément :unresolved s'applique uniquement aux éléments non résolus, pas aux éléments héritant de l'interface HTMLUnknownElement (voir Comment sont gérés les élémentsComment sont gérés les éléments).

```
<style>
  /* appliquer une bordure pointillée aux
éléments non résolus */
  :unresolved {
    border: 1px dashed red;
    display: inline-block;
  }
  /* les éléments x-panel non résolus sont en
rouge */
  x-panel:unresolved {
    color: red;
  }
  /* les éléments x-panel enregistrés sont en
vert */
  x-panel {
    color: green;
    display: block;
    padding: 5px;
  }
</style>

<panel>
  Je suis en noir car le pseudoélément
:unresolved ne s'applique pas à "panel".
  Ce n'est pas un nom valide pour un élément
personnalisé.
</panel>

<x-panel>Je suis en rouge car x-panel:unresolved
s'applique à moi.</x-panel>
```

Pour en savoir plus sur :unresolved, voir A Guide to styling elements sur Polymer. : Lien52

7. Historique et support des navigateurs

7.1. Détection de fonctionnalité

Détecter la disponibilité de l'API revient à vérifier l'existence de document.register() :

```
function supportsCustomElements() {
  return 'register' in document;
```

```
}

if (supportsCustomElements()) {
  // C'est bon, on peut y aller !
} else {
  // Utiliser une solution de remplacement.
}
```

7.2. Support des navigateurs

document.register() a été implémenté dans Chrome 27 et Firefox 23 en activant une option. Cependant, la spécification a quelque peu évolué entre temps et Chrome 31 est le premier à assurer un réel support de la version actuelle de la spécification.

Pour activer la prise en compte des éléments personnalisés dans Chrome 31, tapez chrome://flags/ dans la barre d'adresse et activez l'option « Activer les fonctionnalités expérimentales de Web Platform » (chrome://flags/#enable-experimental-web-platform-features).

En attendant un support plus significatif, vous pouvez utiliser ces solutions de remplacement :

- Polymer (Lien 53), de Google propose une solution (Lien 54) ;
- les x-tags de Mozilla : Lien 55.

7.3. Qu'est devenu HTMLElementElement ?

Ceux qui ont suivi le travail de standardisation savent qu'il a existé la balise <element>. C'était vraiment la panacée. Vous pouviez les utiliser pour enregistrer les nouveaux éléments de façon déclarative :

```
<element name="my-element">
  ...
</element>
```

Malheureusement, il y avait trop de problèmes de performances au niveau du processus de mise à jour, des effets de bord et des scénarios catastrophes qui ont empêché cela de fonctionner correctement. Ainsi, <element> a dû être abandonné. En août 2013, Dimitri Glazkov annonça sa suppression dans un message du public-webapps (Lien 56), en tout cas pour le moment…
Il est toutefois intéressant de noter que Polymer implémente une approche déclarative pour l'enregistrement d'éléments avec <polymer-element>. Comment cela est-il mis en place ? En utilisant document.register('polymer-element') et les techniques présentées dans Créer des éléments à partir d'un gabarit

8. Conclusion

Les éléments personnalisés nous fournissent un outil permettant d'enrichir le vocabulaire du HTML, de lui apprendre de nouvelles choses et d'enrichir les applications Web. En les combinant avec d'autres interfaces comme le DOM fantôme ou <template>, nous commençons à avoir un bel aperçu des composants Web (Web Components : Lien 57). De cette façon, le balisage peut de nouveau être élégant !
Si vous êtes intéressé par les composants Web, je vous invite à regarder du côté de Polymer (Lien 53), qui apporte plus que le strict minimum pour démarrer.
Cet article est la traduction de Custom Elements : defining new elements in HTML (Lien 39) publié sur le site HTML5 Rocks (Lien 58).

Retrouvez l'article d'Eric Bidelman traduit par Didier Mouronval en ligne : Lien 59

L'Internet Rapide et Permanent - La technologie Wi-Fi

Vous disposez d'une connexion permanente et rapide… et maintenant, vous êtes perdu dans la technique…

Cette série « L'Internet Rapide et Permanent », que Christian Caleca nous a aimablement autorisé à reproduire, est là pour répondre à quelques-unes de ces questions. Cet article parlera de la technologie liée aux réseaux WiFi.

1. Les réseaux sans fil

Dans la famille des réseaux « Wireless », ceux qui sont construits selon la famille de normes 802.11 connaissent un énorme développement. Leurs champs d'application les plus communs sont :

- en milieu personnel, le déploiement d'un petit réseau, principalement destiné à partager une connexion internet haut débit ;
- en entreprise, pour permettre la connexion facile de stations de travail nomades (portables) au réseau ou à une partie du réseau de l'entreprise ;
- dans les zones rurales, pour distribuer aux administrés un accès internet obtenu le plus souvent par une solution satellite ;
- dans les lieux publics « hi Tech », pour proposer aux clients munis de portables un accès numérique…

Devant la multitude de solutions proposées, il est probablement nécessaire de faire le point sur cette technologie qui dispose certes d'avantages incontestables, mais qui n'est pas exempte d'inconvénients.
Nous allons essayer de faire le point sur le Wi-Fi, sans entrer trop dans les détails du protocole dans les niveaux 1 et 2, (les couches physiques et liaisons de données, qui sont de l'ordre de la manipulation d'ondes porteuses), ni sur les autres couches d'ailleurs, puisqu'à partir du niveau 3 tout se passe de la même manière que sur un réseau filaire, mais plutôt sur les contraintes de topologie et de sécurité dont il faut absolument tenir compte.

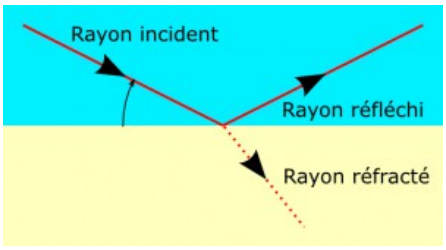
2. Rappels, cela va mieux en le disant…

Il n'est probablement pas inutile de commencer par quelques rappels sur les ondes électromagnétiques.

2.1. Période, fréquence, longueur d'onde…

La « longueur d'onde » fait intervenir une dimension spatiale. Les ondes radio (électromagnétiques) se propagent dans le vide (et dans l'air, avec une erreur négligeable) à la vitesse de 300 000 km/s (3×10^8 m/s). Dans le cas qui nous intéresse, la fréquence est de l'ordre de 2,5 GHz pour les normes 802.11b et 802.11g, les plus utilisées actuellement, ce qui nous donne une période de 4×10^{-10} s. La longueur d'onde est la distance parcourue par l'onde pendant une période, elle est donc ici de l'ordre de 12 cm ($3 \times 10^8 \times 4 \times 10^{-10} = 12 \times 10^{-2}$). On admettra qu'un objet peut constituer un obstacle à la propagation d'une onde lorsque cet obstacle atteint une dimension supérieure ou égale à la longueur de l'onde.

2.2. Les ondes et les obstacles



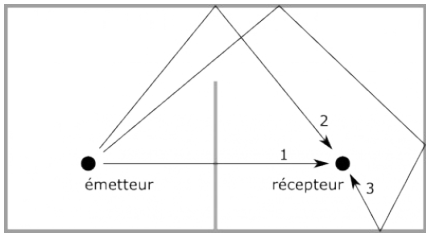
Lorsqu'une onde rencontre un obstacle, sauf si cet obstacle dispose de caractéristiques très particulières, cette onde est en partie réfléchie (renvoyée par l'obstacle dans une autre direction), réfractée (une partie de l'onde traverse l'obstacle) et absorbée (l'obstacle absorbe une partie de l'énergie de l'onde). Les cas particuliers sont :

- l'obstacle réfléchissant, qui fait que la quasi-totalité de l'onde incidente est réfléchie ;
- l'obstacle absorbant, qui fait que la quasi-totalité de l'énergie de l'onde est absorbée.

Il est assez facile d'observer ces phénomènes dans le domaine acoustique. Les ondes ne sont plus électromagnétiques, mais subissent tout de même les effets de réfraction, de réflexion et d'absorption. Nous verrons plus loin ce que ça donne dans une enceinte close.

2.3. Les « échos »

En atmosphère libre (sans obstacles), il n'y a généralement pas de problèmes encore qu'une atmosphère libre n'est que théorique, sauf éventuellement dans l'espace. En effet, sur terre, nous avons au moins le sol qui constitue un obstacle. Généralement, le Wi-Fi s'utilise dans des murs et là, il y a plein d'obstacles.



Imaginons un émetteur et un récepteur placés dans des salles contigües. L'émetteur émet dans toutes les directions, si bien qu'il y aura une multitude d'ondes réfléchies, dont certaines atteindront le récepteur. Dans l'exemple, l'onde 1 atteint directement le récepteur, en traversant la cloison, l'onde 2 l'atteint après une réflexion, l'onde 3 après trois réflexions… Clairement, il y en aura beaucoup plus, avec des chemins différents (plus ou moins longs) et avec des atténuations plus ou moins importantes. Pour une seule source d'émission, le récepteur va recevoir plusieurs fois la même information, plus ou moins atténuée et plus ou moins décalée dans le temps. En acoustique, le problème est bien connu sous le nom de « réverbération » (beaucoup d'échos, avec des décalages temporels très petits).
De plus, en un point donné, deux ondes peuvent parvenir en opposition de phase. Elles n'auront probablement pas la même amplitude, mais leur somme mathématique aura tendance à donner un résultat nul, ce qui conduira à une perte de la porteuse, en ce point précis.
Le traitement de la réverbération est une chose complexe à étudier, mais empiriquement, l'on sait bien que jusqu'à un certain point, ce n'est guère gênant pour récupérer l'information, voire ce peut être bénéfique. En revanche, si le « taux de réverbération » devient trop grand, le signal devient inexploitable (effet « cathédrale »).
Pour les ondes électromagnétiques que nous utilisons pour le Wi-Fi, il en va de même. Ceci pour expliquer une faiblesse majeure du système : dans un bâtiment, il est très difficile, voire impossible, de prévoir la position optimale du ou des émetteurs en fonction des points d'écoute souhaités. Dans la plupart des cas, il faudra procéder à des tests pour obtenir la couverture désirée.
Et pour que les choses soient tout à fait claires, ayez présent à l'esprit que les réseaux Wi-Fi permettent le passage de données dans les deux sens. Autrement dit, ici, chaque point est à la fois émetteur et récepteur, qu'il s'agisse d'une borne d'accès ou d'un poste du réseau.

2.4. Les canaux d'émission

Nous verrons pourquoi plus tard, les normes 802.11xx utilisent des bandes de fréquences divisées en plusieurs canaux. Chaque canal correspond à une fréquence de porteuse bien définie et chaque canal est éloigné de ses voisins par un écart constant en fréquence.
Par exemple, dans les normes 802.11b et 802.11g, il y a en France 13 canaux possibles, de 2,412 GHz à 2,472 GHz, espacés les uns des autres de 5 MHz.
Chaque canal utilise une certaine bande de fréquence (largeur du canal, due à la modulation de la porteuse). La largeur de chaque canal est de 22 MHz, si bien que les canaux se recouvrent. Nous verrons que ceci aura une grande importance dans la suite.

Canal 802.11b ou g	Fréquence centrale	Plage de fréquence ±11 MHz
1	2.412 GHz	2.401 -> 2.423 GHz
2	2.417 GHz	2.406 -> 2.428 GHz
3	2.422 GHz	2.411 -> 2.433 GHz
4	2.427 GHz	2.416 -> 2.438 GHz
5	2.432 GHz	2.421 -> 2.443 GHz
6	2.437 GHz	2.426 -> 2.448 GHz
7	2.442 GHz	2.431 -> 2.453 GHz
8	2.447 GHz	2.436 -> 2.458 GHz
9	2.452 GHz	2.441 -> 2.463 GHz
10	2.457 GHz	2.446 -> 2.468 GHz
11	2.462 GHz	2.451 -> 2.473 GHz
12	2.467 GHz	2.456 -> 2.478 GHz
13	2.472 GHz	2.461 -> 2.483 GHz

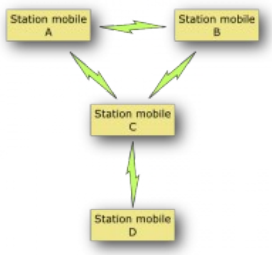
2.5. Et la qualité du matériel ?

Elle a bien entendu son importance. Par exemple, nous savons tous qu'avec deux oreilles, on entend mieux qu'avec une seule. Pas seulement grâce au repérage spatial que l'écoute binaurale permet, mais aussi parce que le cerveau met en œuvre des techniques de corrélation entre les signaux reçus par chaque oreille, qui permettent d'éliminer, jusqu'à un certain point, les perturbations apportées par la réverbération et le bruit.
Les systèmes Wi-Fi peuvent être équipés de techniques similaires, qui permettent plus ou moins efficacement de traiter un signal entaché de réverbération.
Nous n'entrerons pas davantage dans la définition des normes 802.11xx, de toute façon, il n'est pas possible d'agir sur le protocole, de même qu'il n'est pas possible d'agir au niveau 1 d'un réseau Ethernet. Ce qu'il est important de comprendre ici, c'est que les problèmes de propagation sont importants et peuvent considérablement influer sur le résultat obtenu.

3. Architectures, les divers modes de fonctionnement

Il y a fondamentalement deux façons de faire fonctionner un réseau Wi-Fi, suivant ce que l'on souhaite faire.

3.1. Le mode « ad hoc »

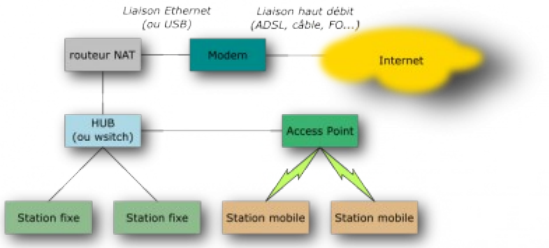


Il n'y a pas dans le réseau de point émetteur/récepteur ayant un rôle particulier. C'est typiquement le mode que l'on choisira si l'on souhaite juste faire communiquer entre elles deux ou trois machines disposant chacune d'une interface Wi-Fi. C'est un mode de fonctionnement rudimentaire, qui peut rapidement devenir compliqué si le nombre de machines en réseau augmente. Chaque station ne peut communiquer qu'avec les stations qui sont à portée. Dans l'exemple : * la station C peut communiquer avec toutes les autres stations, * les stations A, B et C peuvent communiquer entre elles, * la station D ne peut communiquer qu'avec la station C. En aucun cas, la station C ne pourra servir de relais pour que, par exemple, D puisse communiquer avec A. Cet exemple le montre clairement, ce type de réseau n'a d'intérêt que pour permettre à des machines proches (et peu nombreuses) de

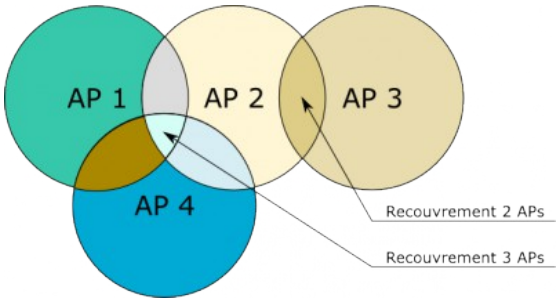
communiquer entre elles en dehors de toute structure.

3.2. Le mode « infrastructure »

Dans ce mode, il y a au moins un émetteur/récepteur Wi-Fi qui joue un rôle particulier, celui de point d'accès (Access Point). C'est typiquement le mode utilisé lorsque l'on souhaite étendre un réseau câblé, genre Ethernet, avec une couverture Wi-Fi pour les portables, ou pour les machines que l'on ne souhaite pas câbler.
Les « modems Wi-Fi », entendez par là les modems ADSL ou câble, qui proposent une connectivité Wi-Fi fonctionnent généralement dans ce mode. C'est ce mode que nous étudierons plus en détail.
En voici une représentation typique :



Ici, toutes les fonctions sont distinctes, mais rien n'interdit que les fonctions modem, routeur NAT et point d'accès soient concentrées dans le même boîtier. Dans un tel cas, les stations fixes et mobiles pourront communiquer entre elles, car nous pourrons faire en sorte qu'elles soient sur le même réseau IP (avec tous les risques que ça comporte au niveau de la sécurité).
Le point d'accès agit un peu comme un HUB pour les stations mobiles. Généralement ce point d'accès (que nous appellerons par la suite AP, pour reprendre la terminologie courante) dispose lui-même d'une adresse IP, qui permet de l'administrer à distance par divers moyens (Telnet, mini serveur HTTP, application dédiée).
Ici, l'AP sert de relais non seulement entre les stations mobiles, mais aussi entre les stations mobiles et les stations fixes. Pour une station mobile, tout se passe comme si elle était connectée au réseau local par un fil. S'il y a un serveur DHCP sur le LAN, les stations mobiles peuvent même recevoir leur configuration IP de façon automatique.
Vous l'aurez compris, le mode « ad hoc » n'a d'intérêt qu'occasionnellement, en dehors de toute structure.



Si l'on souhaite obtenir une couverture convenable sur un site donné, il sera probablement nécessaire de placer plusieurs points d'accès. Dans un tel cas, les zones de couverture de plusieurs points d'accès viendront probablement se recouvrir partiellement.
Dans l'exemple donné, AP1, AP2 et AP4 se recouvrent partiellement. AP2 et AP3 également. Si l'on veut éviter des interférences dans ces zones de recouvrement, il faudra prendre quelques précautions. Bien entendu, AP1 AP2 et AP3 ne devront pas utiliser le même canal. Mais rappelez-vous aussi que les canaux se recouvrent partiellement. Non seulement il faudra utiliser des canaux différents, mais en plus, il faudra que ces canaux ne se recouvrent pas. Ça laisse finalement très peu de choix, reportez-vous au tableau vu plus haut, et vous constaterez qu'il n'est matériellement pas possible de placer quatre points d'accès se recouvrant partiellement, avec les canaux autorisés en France.
En revanche, AP3, qui ne recouvre qu'AP2, pourra utiliser le même canal qu'AP1 ou AP4.

4. Mais d'abord… savoir où l'on va…

4.1. Les risques sanitaires

Nous allons mettre en œuvre un réseau sans fil, qui va utiliser des ondes radio. Les ondes radio, nous l'avons vu, sont capricieuses et peuvent :

- ne pas passer là où l'on voudrait qu'elles passent, ce qui peut conduire à multiplier les AP et, accessoirement, la facture (financière) de l'installation ;
- passer là où l'on ne voudrait pas qu'elles passent, et c'est sans doute ce cas le plus dangereux.

En effet, pour fixer les esprits, vous êtes chez vous et vous montez votre réseau Wi-Fi, qui va permettre à toutes vos machines de communiquer entre elles et d'accéder à l'internet par le truchement de votre ADSL2+ super rapide et tout, sans avoir à tirer le moindre câble, ô combien inesthétique.
Oui, mais qu'est-ce qui va empêcher :

- vôtre (vos) voisin(s) de profiter non seulement de votre accès internet super très haut débit, mais aussi des ressources que vous partagez sur votre réseau local ? ;
- les « geeks » qui, dans leur voiture, passent leur temps à repérer les AP à portée de rue pour en extraire le maximum ?

Nous verrons qu'il y a des moyens de se protéger, et aussi, hélas, des moyens de passer outre ces protections…
Cependant, il faut bien l'admettre, lorsque l'on utilise un portable, le Wi-Fi, c'est bien pratique.

4.2. Le vocabulaire

Pour bien faire du Wi-Fi, il faut déjà maîtriser le vocabulaire qui va avec. Ne vous y trompez pas, monter un réseau Wi-Fi, à moins que vous n'ayez une chance pas possible, ne se fera pas sans mal. La loi de Murphy (Lien 60) et ses innombrables corollaires feront que vous devrez vous battre implacablement avec ce système.
Les documentations, souvent en anglais, usent et abusent de surcroît d'une panoplie d'acronymes incompréhensibles par le non-initié. Un petit exemple ?

Wireless	Commençons par quelque chose de simple, c'est juste un mot anglais qui signifie : sans fil.
WLAN	Comme Wireless LAN : réseau local sans fil.
Wi-Fi	Le nom **Wi-Fi** (contraction de Wireless Fidelity, parfois notée à tort WIFIPhilippe DUVAL2013-12-18T11:51:21.88Quelle différence... ?Raymond 2013-12-18T12:26:36.81Répondre à Philippe DUVAL (18/12/2013, 11:51): "..."Oui, quelle différence ?Et maintenant, c'est mieux:-)Quel oeil) correspond initialement au nom donné à la certification délivrée par la Wi-Fi Alliance (Lien 61), anciennement WECA (Wireless Ethernet Compatibility Alliance), l'organisme chargé de maintenir l'interopérabilité entre les matériels répondant à la norme 802.11. Par abus de langage (et pour des raisons de marketing) le nom de la norme se confond aujourd'hui avec le nom de la certification. Ainsi un réseau Wi-Fi est en réalité un réseau répondant à la norme 802.11. (Définition tirée de l'excellent site CCM (Lien 62)).
Hot Spot	Zone publique couverte par un réseau sans fil (gares ferroviaires, aéroports, hôtels…).
WA	Comme Wireless Adapter. Comprenez : interface réseau sans fil.
AP	On l'a déjà vu : Access Point. Point d'accès en mode « infrastructure ».
BSS	Comme « Basic Service Set ». Ensemble de services de base. Ensemble formé d'un point d'accès et des stations qui y sont connectées. Aussi appelé « Cellule ».
BSSID	Identifiant d'un BSS. Absolument fondamental, comme nous le verrons plus loin, c'est en réalité l'adresse MAC du point d'accès.
ESS	Comme nous l'avons vu, dans un réseau de type « infrastructure », il se fera souvent qu'un seul AP ne suffise pas à assurer la couverture souhaitée. Dans ce cas, il faudra ajouter d'autres points d'accès, en prenant soin qu'ils puissent travailler en bonne intelligence. Si l'on y arrive, on aura alors créé un ESS : Extended Service Set, qui n'est finalement qu'un ensemble homogène de BSS. Cet ensemble, par extension, si je puis dire, peut se rapporter à un seul point d'accès.
ESSID	Identifiant de l'ESS. Il s'agit d'un nom que l'administrateur va donner au(x) point(s) d'accès qui constitue(nt) l'ESS. Dans le cas particulier où l'ESS est constitué d'un seul BSS, on pourra aussi parler de SSID (voyez comme ça devient vite amusant, de parler le Wi-Fi).
DS	Distribution System. Système de distribution qui est là pour connecter entre eux plusieurs AP afin qu'ils puissent constituer un ESS.
IBSS	Independant BSS. Même chose qu'un BSS, mais sans AP. Autrement dit, c'est un ensemble de stations connectées en mode « ad hoc ». Un IBSS doit disposer d'un SSID unique.
WEP	Wired Equivalent Privacy. Système de chiffrement du réseau (au niveau liaison, c'est-à-dire au niveau 2) dont l'ambitieux objectif est de rendre un réseau sans fil aussi sûr qu'un réseau filaire. Vu du côté utilisateur, WEP se résume à une clé de chiffrement qu'il faudra partager entre les partenaires d'un même ESS.
WPA	Wi-Fi Protected Access. Comme WEP n'a pas vraiment convaincu tout le monde, la « Wi-Fi Alliance » (pas besoin de traduire, je pense), a décidé de normaliser un nouveau système de sécurité. Je ne résiste pas au plaisir de citer un fragment d'article paru sur journaldunet.com : Absente du WEP, l'authentification fait son apparition au sein de WPA. Le protocole prévoit deux modes d'authentification : un mode « entreprise » - qui implique d'installer un serveur central (de type Radius par exemple) pour identifier toute personne souhaitant se connecter - et un mode « personnel ». WPA s'appuie sur la famille 802.1x et le protocole EAP (Extensible Authentification Protocol), extension du protocole PPP (Point-to-Point Protocol), qui peut supporter de nombreux mécanismes d'authentification tels que des cartes à jeton, des mots de passe à usage unique et l'authentification par clé publique utilisant des cartes à puce. Vous le voyez, c'est très facile et à la portée du non-initié. Mais nous aurons l'occasion d'en reparler.
MIMO	Multiple-Input Multiple-Output, encore appelé « multipath ». C'est une technologie très complexe, utilisant plusieurs antennes pour diffuser les ondes radio. Pratiquement, l'objectif est d'améliorer les performances en corrigeant autant que possible les problèmes d'interférences. Le résultat annoncé est une minimisation des points d'ombre et une augmentation de la portée. Existe pour 802.11b et 802.11g et devrait être normalisé dans 802.11n. Je n'ai pas pu tester cette technologie, mais en fonction de ce qui en est dit çà et là, il semblerait vivement déconseillé de l'utiliser tant qu'elle ne sera pas normalisée.

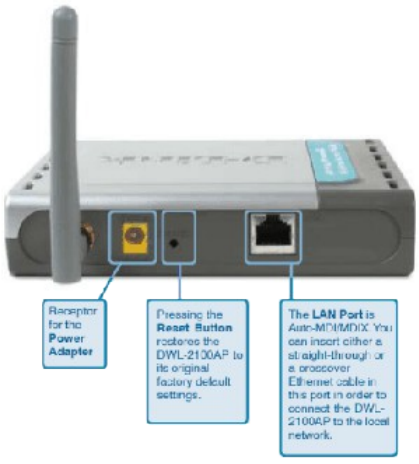
Ce petit lexique devrait nous permettre, du moins dans un premier temps, d'aller un peu plus loin dans la découverte d'un réseau Wi-Fi.

5. Le matériel

Pour poursuivre cet exposé, nous allons nous appuyer sur un minimum de matériel. En principe, le choix du matériel n'a pas une importance capitale pour la mise en œuvre des concepts, encore que chaque constructeur peut ajouter ses

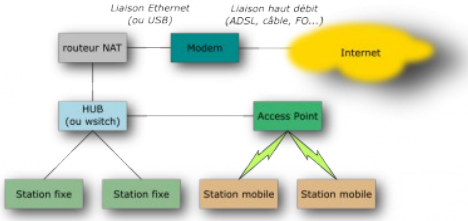
petits « + » à la norme. Méfiez-vous de ces petits « + » qui, s'ils sont utilisés, risquent de provoquer des incompatibilités entre matériels de marques différentes.

Nous disposons d'un réseau filaire Ethernet, nous utilisons bien entendu TCP/IP, le réseau utilise la plage IP 172.16.0.0/16.
Une passerelle vers l'internet est présente (machine Linux avec IPtables), il y a sur ce réseau un serveur DHCP qui permet de configurer automatiquement les hôtes du réseau.
Tout ceci fonctionne parfaitement, et nous voulons maintenant étendre ce réseau avec un accès Wi-Fi pour que les stations portables puissent accéder simplement à ce réseau.
Nous allons donc ajouter un AP (Point d'accès) qui sera la borne DWL-2100AP. Soyons bien clair tout de suite, il n'est absolument pas question de faire de la publicité pour cette marque, ni pour ce modèle (ni de la contre-publicité, d'ailleurs). C'est le matériel dont je dispose, voilà tout.

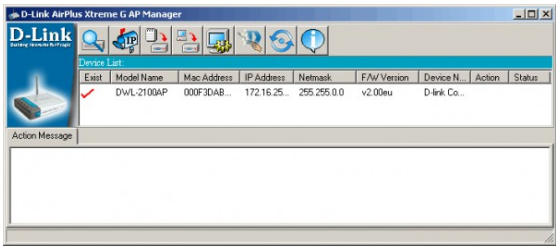


Il s'agit d'un « simple » point d'accès, au sens où il n'y a ni modem ni routeur intégré dans ce boîtier. Il est typiquement conçu pour être connecté à un réseau filaire existant. La face arrière nous montre l'extrême simplicité de ce genre d'équipement. De gauche à droite :

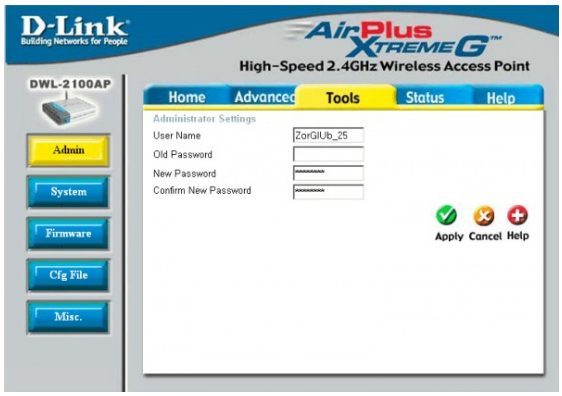
- l'antenne ;
- la prise d'alimentation ;
- le bouton « reset » ;
- la prise réseau RJ 45.



Presque, il suffit de brancher pour que ça marche. Presque, parce que, comme nous allons le voir, cet AP dispose d'une adresse IP par défaut égale à 192.168.0.50.
C'est bien, mais ça ne va pas avec le plan d'adressage que nous avons dans l'exemple. Ici, le seul fait de brancher ne constitue que la première étape, mais pour l'instant, la borne demeure inutilisable.
La borne est fournie avec un utilitaire qui doit être capable de résoudre ce problème :



Une fois le problème de l'adresse IP résolu, nous pourrons nous passer de cet utilitaire, le point d'accès embarquant un mini-serveur web qui nous permettra de le configurer. Est-ce un avantage ? Ce n'est pas si sûr. Par le fait, toute station connectée au réseau pourra accéder à ce serveur web.
Bien entendu, il y a tout de même une identification nécessaire. Par défaut, l'utilisateur s'appelle « admin » et le mot de passe est vide. La première chose à faire est donc de modifier tout ça :



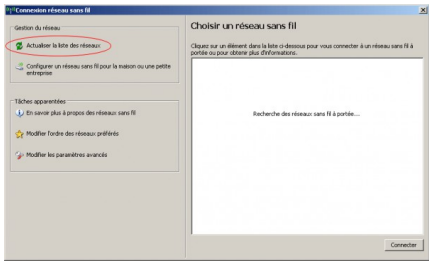
Utilisez, si votre matériel le permet, un nom d'administrateur qui sorte un peu de l'ordinaire et surtout, mettez un mot de passe **solide** et même, pourquoi pas un nom d'utilisateur non trivial (on peut faire largement moins trivial que dans l'exemple), lettres majuscules et minuscules, chiffres, symboles et malgré tout, pensez que vous êtes en HTTP et que le login sera facilement récupérable avec un sniffeur sur le réseau.
Vous me direz alors, « peut-être qu'il serait plus sûr d'exploiter l'utilitaire vu plus haut ? »
AP Manager utilise le protocole tftp il n'y a pas de nom d'utilisateur et le mot de passe circule là aussi en clair.
La troisième solution, ce serait Telnet, mais là encore, les informations sur le login passent en clair…
Vous le voyez, côté réseau local, ce n'est déjà pas terrible du côté de la sécurité. Lorsque vous êtes sur votre réseau domestique, ça peut encore passer, mais sur un réseau d'entreprise, c'est beaucoup plus délicat.
Du côté des clients, j'ai utilisé quatre cartes Wi-Fi différentes :

- une D-Link DWL-G650 802.11g (PCMCIA), la seule des quatre à supporter le système WPA ;

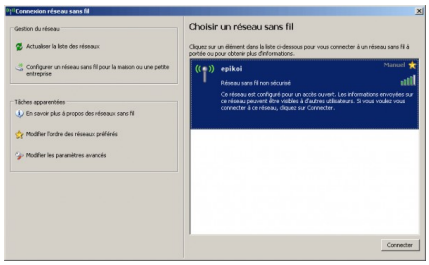
- une D-Link DWL-650 802.11b, la seule que j'aie pu faire fonctionner correctement sous Linux avec les drivers Hostap ;
- une BeWAN PCMCIA 802.11b ;
- une Marvell 802.11g intégrée à la carte mère Asus P5GD2.

6. Config simple, soyons ingénu

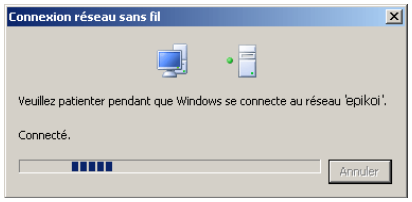
Dans le but de mettre en place une solution qui fonctionne, nous n'allons pas nous poser trop de questions, du moins au début.
Nous supposons que notre point d'accès est correctement configuré côté réseau local, avec une adresse IP accessible pour l'administration. Il suffit alors de configurer la borne en point d'accès, de lui attribuer un SSID, éventuellement un canal. Si nous avons un serveur DHCP sur le réseau local, il ne sera pas nécessaire de configurer celui qui est présent sur la borne. Sinon, ce sera plus simple pour les clients d'utiliser le serveur DHCP de la borne, qu'il faudra alors configurer.
En l'absence de précautions particulières, ça devrait fonctionner tout seul.
Vous prenez un portable équipé Wi-Fi, vous faites une recherche des réseaux Wi-Fi disponibles :



Normalement, s'il est à portée de votre borne, il doit le trouver facilement et vous indiquer le SSID du réseau :



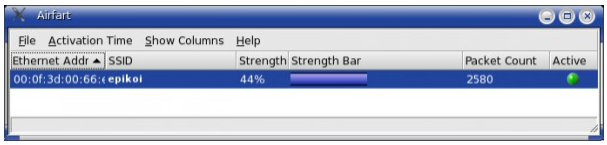
Il vous reste à vous y connecter :



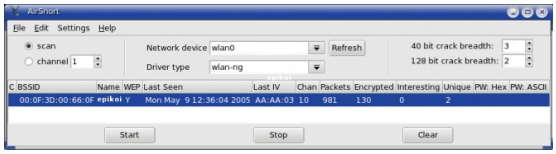
Et ça devrait fonctionner.

6.1. Oui mais…

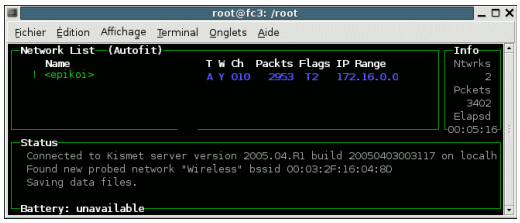
Simple, n'est-ce pas ? Seulement voilà. N'importe qui peut faire ça. Pensez qu'un réseau Wi-Fi passe là où il veut (et pas forcément où vous voulez).
Voici un petit utilitaire graphique, sous Linux, qui fait en gros la même chose :



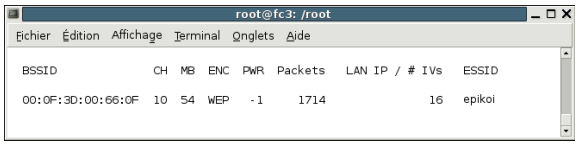
Et, dans la foulée, un autre utilitaire plus puissant, que nous retrouverons d'ailleurs un peu plus loin :



Vous en voulez d'autres ?
Kismet :



Airodump :



Ce dernier est probablement le plus intéressant des trois, mais il y en a encore d'autres… Nous anticipons un peu sur la suite, la capture d'écran est faite avec une protection WEP mise en place, que nous verrons plus loin.
Dans un cas comme dans l'autre, vous voyez que l'indiscret dispose de tous les éléments nécessaires pour se connecter à votre réseau Wi-Fi.

6.2. Et alors ?

Alors, répétons-le, n'importe qui, à portée de votre borne, peut se connecter à votre réseau Wi-Fi et peut par exemple :

- consommer votre bande passante ;
- s'intégrer à votre réseau local et profiter de vos ressources partagées ;
- prendre possession de vos machines pour y installer toutes sortes de choses qui pourront lui servir, par la suite, depuis l'internet ;
- utiliser votre connexion internet pour se livrer, sous votre identité, à toutes sortes d'activités répréhensibles.

Cette liste n'est bien entendu nullement exhaustive.

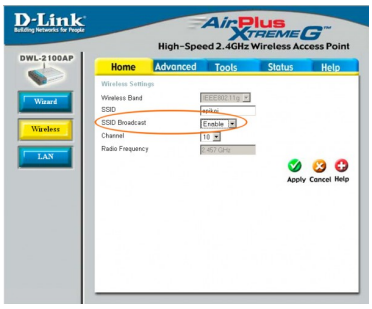
6.3. Donc…

Il n'est clairement pas pensable de rester dans cet état. Il vous faudra verrouiller quelque peu votre installation pour essayer de limiter les risques d'intrusion.

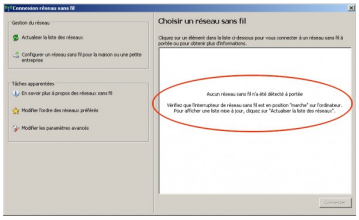
7. Config avancée

7.1. Rendre le SSID « invisible »

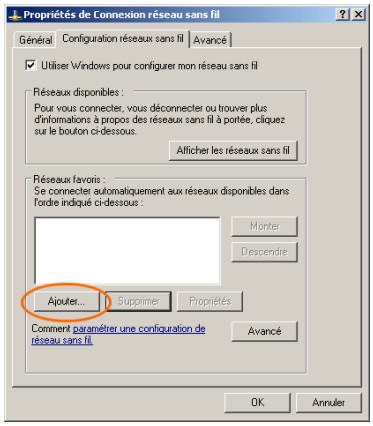
Par défaut, un point d'accès Wi-Fi publie son SSID (en Broadcast). C'est nécessaire pour la découverte de réseaux Wi-Fi accessibles. Mais est-ce vraiment nécessaire, justement ?
Sur un « hot spot » public (il faudrait dire : zone ASFI), c'est probablement nécessaire, puisque l'accès est volontairement public. Sur votre installation personnelle, c'est complètement inutile, il suffit en effet de l'indiquer à vos utilisateurs.
Tous les points d'accès ne savent pas forcément inhiber cette fonction de publication. C'est cependant le cas sur la borne que nous utilisons pour cet exposé :



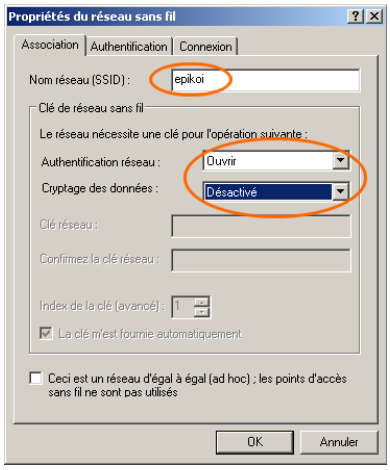
Il suffit de passer ce paramètre à « Disable ». (Vous disiez « parlons français » ?)
Seulement voilà. Si maintenant, nous essayons d'actualiser la liste des réseaux sans fil, il se produit ceci :



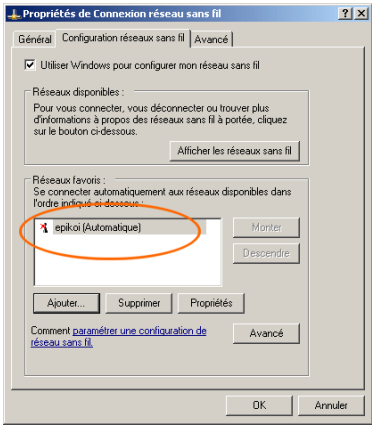
Et là, comment faire ? L'un des moyens les plus simples, consiste :
à cliquer sur « Modifier l'ordre des réseaux préférés » :



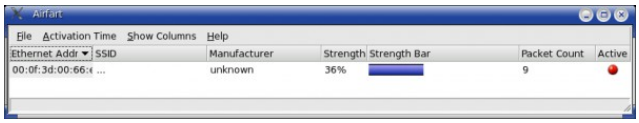
puis de faire « ajouter » :

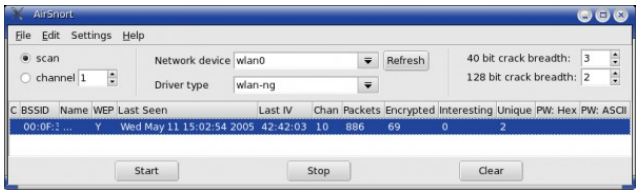


Vous indiquez alors le nom du réseau (SSID), pour l'instant, nous n'avons ni authentification du réseau ni chiffrement des données. Faites « OK », et vos réseaux favoris seront alors à jour.



Vous devriez maintenant pouvoir vous connecter au réseau sans fil.
Voyons ce que disent Airfart et Airsnort :





Chouette ! le SSID n'apparait plus. S'il n'apparait plus, le pirate ne pourra plus se connecter sur le réseau Wi-Fi.
C'est avec ce genre de raisonnement que l'on se retrouve vite à poil (sauf le respect que je vous dois).
Airsnort ou Airodump, savent enregistrer leurs logs au format « pcap » (même format qu'Ethereal, que vous devez commencer à connaitre). Nous n'allons pas déballer tout un log, mais juste une ou deux trames capturées, lorsqu'un client « officiel » se connecte au réseau Wi-Fi :

```
Frame 158 (49 bytes on wire, 49 bytes
captured)
...
IEEE 802.11 wireless LAN management frame
```

```
...
    Tagged parameters (25 bytes)
        Tag Number: 0 (SSID parameter set)
        Tag length: 3
        Tag interpretation: epikoi
        Tag Number: 1 (Supported Rates)
        Tag length: 4
        Tag interpretation: Supported rates:
1,0(B) 2,0(B) 5,5(B) 11,0(B) [Mbit/sec]
        Tag Number: 50 (Extended Supported Rates)
        Tag length: 8
        Tag interpretation: Supported rates: 6,0
9,0 12,0 18,0 24,0 36,0 48,0 54,0 [Mbit/sec]
        Tag Number: 255 (Reserved tag number)
        Tag length: 255
[Malformed Packet: IEEE 802.11]

No.     Time        Source
Destination           Protocol Info

    160 14.124925   D-Link_ab:66:e8
172.16.0.2            IEEE 802.11 Probe
Response[Malformed Packet]

Frame 160 (75 bytes on wire, 75 bytes captured)
...
IEEE 802.11 wireless LAN management frame
...
    Tagged parameters (39 bytes)
        Tag Number: 0 (SSID parameter set)
        Tag length: 3
        Tag interpretation: epikoi
        Tag Number: 1 (Supported Rates)
        Tag length: 8
        Tag interpretation: Supported rates:
1,0(B) 2,0(B) 5,5(B) 11,0(B) 6,0 12,0 24,0 36,0
[Mbit/sec]
        Tag Number: 3 (DS Parameter set)
        Tag length: 1
        Tag interpretation: Current Channel: 10
        Tag Number: 7 (Country Information)
        Tag length: 6
        Tag interpretation: Country Code: GB, Any
Environment, Start Channel: 1, Channels: 13, Max
TX Power: 20 dBm
        Tag Number: 42 (ERP Information)
        Tag length: 1
        Tag interpretation: ERP info: 0x4 (no
Non-ERP STAs, do not use protection, short or
long preambles)
        Tag Number: 50 (Extended Supported Rates)
        Tag length: 4
        Tag interpretation: Supported rates: 9,0
18,0 48,0 54,0 [Mbit/sec]
        Tag Number: 255 (Reserved tag number)
        Tag length: 255
...
```

Peu importent les autres paramètres (que j'ai laissés parce qu'ils ne sont pas totalement inintéressants), ce qui compte ici, c'est que l'on peut malgré tout parfaitement capturer le SSID du réseau Wi-Fi.
D'ailleurs, en laissant tourner Airsnort ou Airodump, sitôt qu'un client viendra se connecter au réseau Wi-Fi, le SSID apparaitra.
Cette précaution n'arrêtera donc que les parfaits profanes qui ne savent pas aller au-delà de la découverte d'un SSID dans les trames de broadcast. C'est donc une protection finalement assez illusoire.

7.2. N'autoriser que les adresses MAC connues

La plupart des points d'accès permettent de n'autoriser la connexion Wi-Fi qu'à une liste d'adresses MAC connues. Une adresse MAC, c'est dépendant du hardware de l'interface. Vous connaissez vos interfaces et vous n'autorisez qu'elles. L'indiscret, qui arrivera avec son portable à portée de votre réseau, même s'il arrive à découvrir votre SSID, ne sera pas accepté parce que son adresse MAC ne figurera pas dans la liste des adresses connues.
Soit, mais :

- cette liste est difficilement gérable lorsqu'elle devient un peu longue ;
- la plupart des Linuxiens le savent bien, une adresse MAC, ça peut se changer.

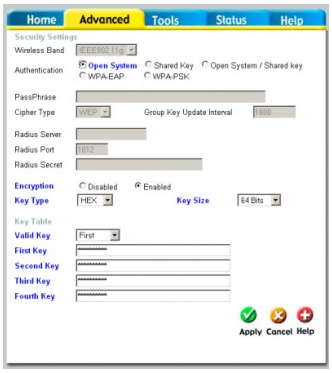
Notre pirate sniffe votre réseau. Bien entendu, lorsqu'un client « connu » se connecte, il devient alors très facile de lire son adresse MAC :

```
Frame 158 (49 bytes on wire, 49 bytes
captured)
    Arrival Time: May 11, 2005 16:00:27.146581000
...
IEEE 802.11
    Type/Subtype: Probe Request (4)
    Frame Control: 0x0040 (Normal)
    Duration: 0
    Destination address: ff:ff:ff:ff:ff:ff
(Broadcast)
    Source address: 00:11:2f:41:cd:a1
(172.16.0.2)
    BSS Id: ff:ff:ff:ff:ff:ff (Broadcast)
...
```

Une fois que l'on dispose d'une adresse MAC valide et du SSID, je vous laisse chercher sur les sites spécialisés, quelles sont les multiples façons de s'introduire quand même sur votre réseau « protégé ».
Pour l'instant, nous n'avons pas réussi à mettre en place des protections bien efficaces. De plus, les connexions n'étant pas chiffrées, même sans se connecter à votre réseau, des « sniffeurs » Wi-Fi permettront aisément de voir vos mots de passe qui circulent en clair (POP, IMAP, FTP…), et même, suivant le cas, des données que vous aimeriez bien pouvoir garder pour vous.

8. WEP, une pincée de chiffrement

Nous allons avoir recours aux bonnes vieilles techniques de chiffrement des données, pour essayer de sécuriser un peu plus notre réseau sans fil.



WEP utilise le protocole de chiffrement RC4 et une clé de chiffrement symétrique et statique. Nous n'entrerons pas dans les détails de RC4, ce qui est du ressort des spécialistes de la cryptographie, disons simplement qu'il a été démontré que ce protocole comportait quelques faiblesses graves.
La clé statique doit être distribuée manuellement sur le ou les points d'accès ainsi que sur tous les clients du réseau sans fil. Nous savons qu'un secret partagé ne reste jamais secret bien longtemps, et plus il est partagé, plus son temps de vie est court.
Nous n'avons pas encore commencé, que déjà, trois grosses faiblesses sont identifiées dans le système WEP :

- le fait que le secret est partagé, ce qui introduit une faiblesse dudit secret ;
- le mode de distribution de la clé statique (qui n'est renouvelée que par le bon vouloir de l'administrateur du réseau) ;
- le protocole de chiffrement lui-même, qui est réputé faible.
  Le second point est souvent renforcé par les équipementiers en proposant d'introduire non pas une seule clé, mais plusieurs, en indiquant simplement l'index de la clé actuellement utilisée. Autrement dit, l'administrateur pourra configurer les différents nœuds du réseau avec un ensemble de plusieurs clés (quatre dans le cas du point d'accès utilisé ici) en indiquant quelle est actuellement la clé utilisée. C'est un complément de protection qui ne vaut que si la clé active est souvent changée, et de façon aléatoire, autant que possible. Mais cette procédure de changement de clé active reste manuelle et doit être exécutée sur tous les nœuds du réseau.

La longueur de la clé peut actuellement être de 64 bits ou de 128 bits, ce qui est à la fois vrai et faux. RC4 nécessite un « vecteur d'initialisation » de 24 bits, qui est généré par RC4 lui-même, si bien que la clé définie par l'administrateur ne fera en réalité que 40 ou 102 bits.
La clé partagée peut également servir à faire une pseudoauthentification des nœuds sur le réseau. Si l'on choisit l'authentification nommée « shared-key », lorsqu'un client va essayer de se connecter sur un point d'accès, ce dernier lui enverra un texte en clair, que le client chiffrera et renverra au point d'accès.
Ainsi, le point d'accès pourra vérifier que le client dispose bien de la clé et le client, s'il est accepté par le point d'accès en déduira que l'AP dispose de la même clé que lui.
Ainsi également, tout indiscret qui regarde ce qu'il se passe sur votre réseau pourra disposer d'un texte en clair et de son équivalent chiffré, ce qui est une information de premier ordre pour découvrir la fameuse clé. Il faut donc éviter d'utiliser cette méthode d'authentification.
Pour le reste, RC4 va faire son travail et va chiffrer les trames qui circulent sur le réseau sans fil, au niveau 2 (transport).

8.1. Pratiquement…

Un intrus, même s'il découvre votre ESSID, même s'il usurpe une de vos adresses MAC autorisées, ne pourra pas se connecter à votre réseau tant qu'il ne disposera pas de la clé partagée.

Donc, nous avons résolu le problème, mais à certaines conditions. En effet, quels sont les moyens dont le pirate dispose, pour s'emparer de la fameuse clé ?

8.1.1. Le « social engineering »

Comme son nom l'indique, cette méthode est purement « sociale ». Elle consiste à obtenir l'information par un membre quelconque de l'organisation, qui partage le secret. Un exemple ? Un ami, une relation, un collaborateur occasionnel ont eu besoin d'utiliser votre réseau Wi-Fi une fois et vous leur avez communiqué la clé.

8.1.2. La cryptanalyse

Ici, nous entrons dans le domaine informatique. L'indiscret se poste en un point où il peut capturer les trames de votre réseau. Il va alors enregistrer sur sa machine tout ce qu'il capte, pour y découvrir les points faibles du système RC4 appliqué à la norme 802.11. Un outil comme Airodump fait ça très bien, et indiquera en temps réel le nombre de vecteurs d'initialisation (IV) susceptibles d'aider un autre outil (Aircrack) à découvrir votre clé. On estime que pour une clé de 64 bits (40 en réalité), un échantillon de 200 000 trames représentatives permet de déduire la clé en seulement quelques minutes.
Récupérer l'échantillon représentatif peut prendre un « certain temps », qui dépend principalement du trafic généré sur votre réseau. La technique étant (pour l'instant) purement passive, si votre réseau est peu utilisé (cas par exemple d'un réseau sans fil personnel, qui ne sert qu'à partager une connexion internet pour deux ou trois clients), l'opération peut durer plusieurs semaines. Dans le cas d'un réseau d'entreprise, ça pourra aller naturellement plus vite.
Si notre pirate est pressé, il peut disposer d'outils qui vont artificiellement générer du trafic sur votre réseau sans fil. La technique n'est plus seulement passive et donc plus facilement repérable, encore faut-il que l'administrateur reste très attentif pour la détecter. Dans un tel cas, quel que soit le trafic « naturel » généré sur votre réseau, le pirate réussira probablement à récupérer son échantillon représentatif en une ou deux journées tout au plus.

8.2. Finalement…

WEP ne peut raisonnablement être utilisé que dans des cas très limitatifs et avec de fortes contraintes administratives :

- l'intrusion du réseau ne doit présenter que peu d'intérêt, c'est un élément important qui détermine la motivation de l'intrus. Votre réseau domestique ne sera probablement pas la proie la plus intéressante, ce qui sera certainement moins vrai pour votre réseau d'entreprise ;
- le trafic généré naturellement doit être le plus faible possible ;
- il vaut mieux s'assurer régulièrement que le trafic reste conforme à l'utilisation que vous avez de votre réseau ;
- la clé partagée doit être modifiée le plus souvent possible, puisque dans le pire des cas, on peut estimer que la durée de vie d'une clé à 64 bits n'est que d'environ une journée. Utilisez donc de préférence une clé à 128 bits, qui nécessitera un plus gros échantillon représentatif, ce qui

allongera (un peu) sa durée de vie ;
- le nombre de nœuds doit être le plus faible possible (moins un secret est partagé, moins il risque de fuir).

Vous l'avez compris, WEP est à proscrire en entreprise et n'est utilisable, avec précautions, qu'en environnement domestique.

9. Conclusions

9.1. Alors, on fait comment ?

À l'heure où ces lignes sont écrites, il existe plusieurs déclinaisons de la norme 802.11. Parmi les plus courantes :
- 802.11b. Comme son nom ne l'indique pas, elle est la plus ancienne et la plus courante, c'est celle qui a été utilisée pour cet exposé, elle offre un débit théorique de 11 Mbit/s et utilise la bande de fréquences des 2,4 GHz ;
- 802.11a. Plus rapide (débit théorique de 54 Mbit/s), elle utilise la bande des 5 GHz et n'est pas compatible avec la norme 802.11b ;
- 802.11g. Elle propose également un débit théorique de 54 Mbit/s, mais dans la bande des 2,4 GHz, ce qui lui permet de rester compatible avec les équipements 802.11b.

Toutes ces normes incluent le système de protection WEP, dont on a vu les limites.
Devant les risques encourus, il a été développé un système de chiffrement plus fort, le WPA, qui devrait être inclus dans la norme 802.11i.
WPA est plus « solide » que WEP, mais nécessite pour être le plus efficace une infrastructure lourde, incluant un serveur d'authentification comme un serveur RADIUS (utilisé sur les connexions de type PPP). C'est cependant un moyen nécessaire pour obtenir une sécurité acceptable en milieu professionnel. Pour un milieu personnel, WPA peut tout de même être mis en œuvre sans la présence d'un serveur d'authentification. On se ramène alors à un système proche (dans sa mise en œuvre) du WEP, mais avec un système de chiffrement plus robuste.
WPA est inclus sur les équipements récents à la norme 802.11g, mais pourrait être ajouté aussi aux normes 802.11a et 802.11b.
Pour la petite histoire, WPA en est déjà à sa seconde version, la première ayant été démontrée trop peu fiable quelques jours seulement après son annonce.
Finalement, en milieu professionnel, il est nécessaire d'adopter la méthode la plus efficace, à savoir WPA, avec un serveur d'authentification, ce qui permettra d'authentifier non seulement le poste qui se connecte, mais également la personne qui s'en sert (ce que WEP ne sait pas faire), et permettra un chiffrement difficile à casser.
En milieu personnel, on pourra utiliser du WPA sans serveur d'authentification, avec une « pass phrase » (sorte de mot de passe beaucoup plus long qu'un mot de passe habituel) si tout l'équipement dont on dispose supporte cette méthode, sinon, il faudra se rabattre sur WEP, avec une clé la plus longue possible, et qu'il faudra penser à changer assez régulièrement. Il est trop risqué, même pour un usage domestique, de mettre en place un réseau Wi-Fi sans chiffrement.

9.2. Et MIMO ?

Ça n'a rien à voir avec la sécurité, mais cette technique qui utilise, rappelons-le, plusieurs antennes et plusieurs émissions (sur le même canal ou non, suivant les implémentations), dans le but de gérer au mieux les problèmes liés aux réflexions et aux absorptions des ondes par des obstacles semble très prometteuse, à en croire les divers tests publiés sur ce sujet.
L'inconvénient majeur, à l'heure où ces lignes sont écrites, est que le procédé n'est pas entièrement normalisé et que les solutions proposées par les divers constructeurs ne sont pas forcément compatibles entre elles, pire, elles peuvent même être parfaitement incompatibles.
À moins de s'astreindre à n'utiliser les services que d'un seul constructeur, il semble donc préférable d'attendre une plus grande maturité de cette technologie qui devrait être incluse dans la norme 802.11n.

Retrouvez l'article de Christian Caleca en ligne : Lien 63

Théorie des collisions : Formes 2D simples

Nous allons commencer ici par les algorithmes de collision les plus élémentaires.

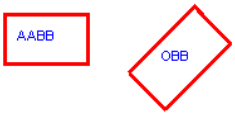
1. Introduction

Nous allons commencer ici par les algorithmes de collision les plus élémentaires.

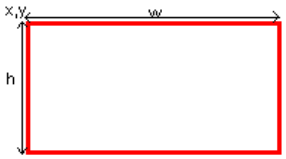
2. Point dans Axis Aligned Bounding Box

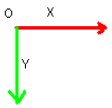
2.1. Définitions

Tout d'abord, définissons une **Axis Aligned Bounding Box** (AABB). Il s'agit d'un rectangle aligné avec les axes, c'est-à-dire que ses côtés sont parallèles aux axes des x et des y de votre repère (de votre écran pour les cas standard).



À la différence d'une **Oriented Bounding Box** (OBB), qui est un rectangle qui peut être orienté : ses côtés ne sont pas obligatoirement parallèles aux axes de votre repère.
Une AABB peut être définie par quatre paramètres : la position x,y de son coin supérieur gauche (en 2D, l'axe Y va vers le bas). Ainsi que de sa largeur w (comme width) et sa hauteur h (comme height).





Cela donne, en C, la structure suivante :

```
struct AABB
{
  int x;
  int y;
  int w;
  int h;
};
```

Notez, pour les utilisateurs de SDL, que cette structure est exactement SDL_Rect (à un type près), et que donc SDL_Rect est parfaite pour décrire une AABB.

2.2. Applications

Ce type de collision cherche donc à savoir si un point (de coordonnées x,y) est dans une AABB ou non.
Dans quel cas avons-nous besoin de ce type de collision ? Par exemple pour un jeu de tir comme Opération Wolf, ou la sélection d'un menu de ce bon vieux Warcraft premier du nom :

Dans l'image à gauche, j'ai encadré les cibles en vert. Ce sont les AABB qui les portent. Bien que cette collision ne soit pas parfaite (vous pouvez descendre l'ennemi en tirant juste au-dessus de son épaule), elle est très utilisée et rapide.
(Certains vont me dire qu'Opération Wolf affine ses collisions… ça se peut, mais considérons que non.)
L'idée de ce jeu est simple : on déplace le curseur à la souris (pointeur rouge) et quand on clique, on regarde s'il est dans une AABB ou non, tout simplement.
Pour la deuxième image, chaque option du menu est un rectangle, une AABB. On va aller cliquer sur une option ou une autre avec la souris. C'est la même collision.

2.3. Calcul de collision

La fonction de collision aura cette signature :

```
bool Collision(int curseur_x,int curseur_y,AABB box)
```

Notes :

- Je renvoie un bool même si celui-ci n'est pas défini en C, vous adapterez si besoin. Ce choix a été fait pour donner davantage de sémantique au code. Vous pouvez définir en C le type et les deux constantes suivantes :

```
typedef int bool;
#define true 1
#define false 0
```

- Idéalement, en C, il est préférable de passer un pointeur vers une structure plutôt que la structure entière, cependant, ce tutoriel voulant rester théorique, je n'alourdirai pas le code par des pointeurs.

Le calcul est très simple : il y a collision si et seulement si le point est à l'intérieur de la boite.
Le point supérieur gauche est : (box.x;box.y)
Le point inférieur droit est : (box.x+box.w-1;box.y+box.h-1)
Pourquoi -1 ? Parce que nous commençons à 0. Si nous ajoutons juste box.w à box.x, nous tombons sur le premier point hors de la boite. Cependant, on peut se passer du -1 si on considère que le point testé sera strictement inférieur à ce premier point après la boite.
Du coup, la fonction de collision est triviale :

```
bool Collision(int curseur_x,int curseur_y,AABB box)
{
   if (curseur_x >= box.x
    && curseur_x < box.x + box.w
    && curseur_y >= box.y
    && curseur_y < box.y + box.h)
       return true;
   else
       return false;
}
```

Note : il m'a été reproché de faire un if avec return true ou false, au lieu de mettre directement la condition dans le return, comme le permet le langage C. Ce choix a été fait pour des raisons de clarté, de sémantique, et de compréhension, car tous les langages ne permettent pas de factoriser de la sorte. Si vous trouvez ça horrible, vous pouvez bien sûr adapter.
Je pense que la fonction parle d'elle-même. Voici donc notre première fonction de collision !

Les bibliothèques graphiques 2D utilisent souvent des nombres entiers (int, short…) pour les coordonnées. On parle de coordonnées discrètes. On dit qu'on est dans le plan N^2. Si vous faites de la 2D avec OpenGL, avec glOrtho (Lien 64), vous utilisez des float pour les coordonnées. On parle de coordonnées réelles, on est dans le plan R^2. Cet algorithme marche aussi bien dans N^2 que dans R^2 (il suffit d'adapter les coordonnées en float).

3. Collision entre deux Axis Aligned Bounding Box

Voici un autre test de collision.
Cette fois-ci, nous allons voir comment tester la collision entre deux AABB.

3.1. Applications

Cette fonction est extrêmement utilisée dans énormément de jeux.

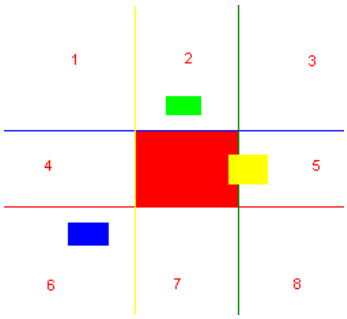
Nous voyons à gauche ce cher Mario. Il a sa boite englobante (en vert), et la tortue aussi. Comment voir s'il la touche ? Eh bien en testant une collision entre deux boites englobantes.
Pareil, à droite, Gradius est un shoot'em up à l'ancienne. Chaque vaisseau, chaque missile ont leur boite englobante. Il faut tester les collisions entre notre vaisseau et tous les vaisseaux et missiles ennemis (ce qui nous fait perdre), et également la collision entre nos missiles et les vaisseaux ennemis (pour les détruire).
Il y a beaucoup de collisions à tester, cela doit donc être rapide de préférence.

3.2. Calcul de collision

La signature de notre fonction sera la suivante :

```
bool Collision(AABB box1,AABB box2)
```

L'idée est la suivante. Regardons le petit schéma ci-dessous :



Le rectangle rouge est box1. J'ai dessiné des traits, rouge, bleu, jaune et vert, en prolongeant les côtés à l'infini. Pour savoir si un autre rectangle touche le rectangle rouge, raisonnons à l'envers : essayons de savoir quand il ne touche pas.
Un rectangle box2 ne touche pas si :

- il est complètement à gauche de la ligne jaune ;
- il est complètement à droite de la ligne verte ;
- il est complètement au-dessus de la ligne bleue ;
- il est complètement au-dessous de la ligne rouge.

Voyons avec le dessin ci-dessous les exemples :

- le rectangle bleu est non seulement complètement à gauche de la ligne jaune, mais aussi complètement au-dessous de la ligne rouge : il ne touche pas ;
- le rectangle vert est complètement au-dessus de la ligne bleue : il ne touche pas ;
- Le rectangle jaune n'est ni complètement en haut, ni à gauche, ni à droite, ni en bas : il touche.

Voici la règle énoncée :

Si la box2 est complètement à gauche, ou complètement en haut, ou complètement à droite, ou complètement en bas, alors elle ne touche pas. Sinon, elle touche.

Pour savoir si la box2 est à droite du trait vert (donc trop à droite), on regarde simplement si sa coordonnée x (son minimum en x) est plus grande que le maximum en x de box1 (le maximum en x étant box1.x + box1.w -1).
Donc on obtient le test suivant :

$$box2.x > box1.x + box1.w - 1$$

Ce qui équivaut à :

$$box2.x >= box1.x + box1.w$$

Pour les quatre autres directions, le calcul est similaire.
La fonction suivante en découle :

```
bool Collision(AABB box1,AABB box2)
{
   if((box2.x >= box1.x + box1.w)      // trop à
droite
    || (box2.x + box2.w <= box1.x) // trop à
gauche
    || (box2.y >= box1.y + box1.h) // trop en bas
    || (box2.y + box2.h <= box1.y))  // trop en
haut
          return false;
   else
          return true;
}
```

La rapidité de cette collision est assurée. En très peu de calculs, on a notre résultat, ce qui permet de pouvoir faire beaucoup de tests dans le jeu sans ralentissements.
Cette collision peut paraître grossière, mais elle est souvent largement suffisante pour beaucoup de cas. D'autres collisions plus fines, mais aussi plus coûteuses en temps de calcul, utiliseront cette collision auparavant, afin d'éliminer facilement les cas où les objets ne se touchent clairement pas.

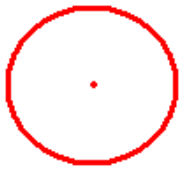
- Notez qu'on se moque de savoir quelle est la box1 et quelle est la box2. Si la box1 touche la box2, alors la box2 touche aussi la box1.
- Ça fonctionne avec des boîtes de tailles différentes, aucune restriction sur ce point.
- Cet algorithme marche aussi bien dans N^2 que dans R^2.

4. Cercles

Les cercles sont également fort intéressants pour les collisions. On peut très rapidement tester si un point est dans un cercle, ou si deux cercles se touchent, ce qui peut être fort utile.

4.1. Définitions

Un cercle, c'est un centre x,y et un rayon.



Nous pouvons immédiatement définir une structure de cercle :

```
struct Cercle
{
   int x,y;
   int rayon;
};
```

4.2. Applications

Imaginons que vous vouliez cliquer dans une zone de cercle (crever des ballons par exemple), ou alors que vous fassiez un jeu de billard ou un jeu du genre Puzzle Bubble (même si je ne suis pas sûr que Puzzle Bubble utilise rigoureusement cette collision), alors cette collision vous sera fort utile.

4.3. Calcul de collision

4.3.1. Point dans un cercle

Tout d'abord, voyons le cas d'un point dans un cercle.
La signature de notre fonction sera la suivante :

```
bool Collision(int x,int y,Cercle C)
```

Vous souhaitez savoir si le point x,y est dans le cercle ou non.
C'est très simple, il suffit de calculer la distance du point x,y au centre du cercle. Si cette distance est supérieure au rayon, alors vous êtes dehors, sinon, vous êtes dedans.
Pour le calcul de distance, pensez à Pythagore. Le calcul de la distance euclidienne (Lien 65) dans un plan se calcule simplement :

$$d = sqrt((x - C.x)^2 + (y - C.y)^2)$$

Le seul inconvénient de cette méthode, c'est qu'il y a une racine carrée. C'est une opération assez coûteuse, même si maintenant, les machines sont suffisamment puissantes pour ne pas trop s'en rendre compte. Si on peut l'éviter, alors on l'évite.
Et dans notre cas, on peut. En effet, on souhaite savoir si d>C.r ou pas. Or, d et C.r étant positifs, on peut dire que :

$$d > C.r <=> d^2 > C.r^2$$

Du coup, la racine carrée disparaît :

$$d^2 = (x - C.x)^2 + (y - C.y)^2$$

La fonction de collision est donc très simple :

```
bool Collision(int x,int y,Cercle C)
{
   int d2 = (x-C.x)*(x-C.x) + (y-C.y)*(y-C.y);
   if (d2>C.rayon*C.rayon)
      return false;
   else
      return true;
}
```

Note : en C, la « puissance 2 » n'existe pas. Je multiplie donc (x-C.x)*(x-C.x), ce qui en soi est un vilain copier/coller, et un calcul fait deux fois. On pourrait éviter ça avec des variables intermédiaires, mais y gagnerait-on en vitesse ? Pas sûr. Une chose par contre est sûre, n'utilisez pas la fonction pow() en C pour faire des carrés, jamais. C'est sortir l'artillerie lourde, et perdre du temps, pour une simple mise au carré.

Note : une idée pour optimiser davantage est de stocker directement le rayon au carré dans la structure du cercle. Si le rayon reste constant, on gagnera en optimisation en évitant à chaque fois de recalculer C.rayon * C.rayon.

4.3.2. Collision de deux cercles

Nous souhaitons maintenant savoir si deux cercles se touchent. Pour un jeu de billard par exemple, c'est fort utile.
La signature de notre fonction sera celle-ci :

```
bool Collision(Cercle C1,Cercle C2)
```

Comment savoir si deux cercles se touchent ? En réalité, c'est très simple : nous mesurons la distance entre leurs deux centres, et il suffit de voir si cette distance est supérieure ou inférieure à la somme des rayons.
La distance entre les rayons sera bien sûr un calcul similaire à ce qu'on a vu au-dessus :

$$d = sqrt((C1.x - C2.x)^2 + (C1.y - C2.y)^2)$$

L'astuce pour éliminer les carrés est la même. Nous obtenons donc la fonction suivante :

```
bool Collision(Cercle C1,Cercle C2)
{
   int d2 = (C1.x-C2.x)*(C1.x-C2.x) + (C1.y-
C2.y)*(C1.y-C2.y);
   if (d2 > (C1.rayon + C2.rayon)*(C1.rayon +
C2.rayon))
      return false;
   else
      return true;
}
```

- Cela fonctionne même avec des cercles de rayons différents. Une bille et une boule de pétanque par exemple.
- Cet algorithme marche aussi bien dans N^2 que dans R^2.

Note : si le rayon des cercles est constant, alors (C1.rayon + C2.rayon) * (C1.rayon + C2.rayon) est constant aussi. On pourrait donc, si besoin, stocker cette valeur quelque part pour optimiser le calcul au lieu de le refaire à chaque fois. Merci à Sylvior pour cette remarque.
Tous ces algorithmes sont très rapides et utiles pour beaucoup de problèmes.

Retrouvez l'article de Fvirtman en ligne : Lien 66

Interview d'un studio de jeux vidéo : Heliceum

Voici la retranscription d'un interview téléphonique réalisé le 7 novembre 2013 avec Sébastien Broussaud, directeur technique chez Heliceum, un studio de développement de jeux et d'applications mobiles parisien.

1. Introduction

1.1. Qui êtes-vous ?

Je suis Sébastien Broussaud, directeur technique chez Heliceum (Lien 67), un studio de développement parisien. Je suis ingénieur en informatique spécialisé en intelligence artificielle. Auparavant, j'ai travaillé à la DGA, puis dans la finance, notamment pour la Société Générale. Il y a environ six ans, j'ai commencé le développement de jeux vidéo chez Dontnod entertainment (Remember Me - 2012) (Lien 68) et depuis environ deux ans et demi j'ai aidé à fonder Heliceum en tant que directeur technique.

1.2. Comment êtes-vous arrivé dans le monde du jeu vidéo ?

En fait, lors de mes études, je m'étais spécialisé en intelligence artificielle, qui est déjà proche du monde du jeu vidéo. À la création de Dontnod, le directeur technique cherchant les premiers développeurs m'a contacté, car nous avions fait nos études ensemble. Chez Dontnod, j'ai aidé au développement en mettant en place une première base technique et à la préproduction du jeu.

1.3. Qu'est-ce que Heliceum ?

C'est donc un studio de développement parisien constitué de 25 employés. La principale activité est le développement de jeux vidéo sur smartphones et tablettes et de quelques applications Internet.

1.4. Comment s'est créé Heliceum ?

Le responsable de la société, Adrien Dassault m'a contacté pour l'aider à fonder la société. Il souhaitait passer d'une société qui ne faisait pas du tout d'informatique à une société de développement de jeux, l'une de ses passions. Il a donc essayé de recruter des gens qui connaissaient bien le milieu, que ce soit dans le domaine du graphisme ou du développement. On a commencé à trois, il y a deux ans et demi pour être maintenant 25.
Au début, le studio ne travaillait que sur de petits projets puis, ils ont monté en gamme avec « Faker$ » (Lien 69), « Human Defense » (Lien 70) et aujourd'hui « Sale Gosse » (Lien 71) ou encore avec des applications Business to Business (B2B) comme « Le grand défi Le Figaro » (Lien 72) ainsi que d'autres petits projets.

Voir la vidéo : Lien 73

1.5. Pouvez-vous me décrire l'équipe d'Heliceum ?

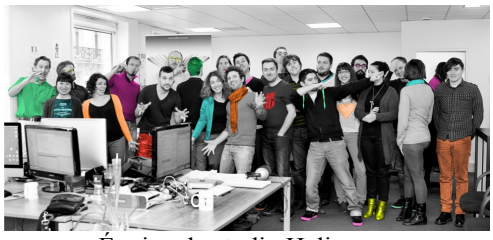
Dans Heliceum, il y a plusieurs corps de métier décomposés en deux branches :

- la branche **production**, créant les applications et les jeux ;
- la branche **marketing**.

Je m'occupe de la branche production, dans laquelle on retrouve plusieurs corps de métier :

- les **game designers**, qui s'occupent du design et des spécifications des applications et jeux ;
- les **graphistes**, produisant les ressources graphiques ;
- le **pôle développement**, produisant tout le code du jeu et s'occupant de tous les aspects techniques.

Il y a aussi un chef de projet et une équipe assurance qualité et test permettant de s'assurer que les applications mises en ligne soient d'une qualité convenable.


Équipe du studio Heliceum

2. Développement des jeux

2.1. Pourquoi « Sale Gosse » n'est-il disponible que sur l'Apple Store, Google Play et Amazon, mais pas sur Windows Market, alors que l'icône est grisée ?

En effet, on n'a pas développé pour Windows Market, mais cela arrivera peut-être dans quelques mois. Heliceum discute avec les équipes techniques de Microsoft et selon la place que Heliceum peut obtenir sur Windows Store, le studio réfléchit à ce portage.

2.2. Comment s'effectue le choix de porter une application ou non ?

Historiquement, les applications de Heliceum étaient uniquement compatibles iOS, car à la création du studio, les seules applications pouvant dégager des revenus n'étaient que sur iOS. Le marché a changé et Android a bien avancé, faisant qu'il est maintenant intéressant de faire des applications iOS et Android. Amazon étant similaire à Google Play, le portage n'est pas coûteux et la seule contrainte a été de refaire le système de monétisation pour accéder aux serveurs d'Amazon au lieu de ceux de Google. Par contre, le portage pour Windows Phone demande plus d'efforts et les technologies ne sont pas les

mêmes, nécessitant plus d'énergie sans savoir si cela va rapporter l'investissement que l'on engage dans cette tâche.

2.3. Quelles technologies utilisez-vous donc pour la création de vos jeux ?

Pour « Sale Gosse », nous utilisons Unity. La partie technique et la partie commune vont donc marcher sur Windows Phone. Le problème, c'est que l'on va devoir remettre tout ce qui est bibliothèques externes et les recoder pour les passer sur Windows Phone. Finalement, il y a pas mal de choses que l'on utilise qui ne vont pas marcher, mais qui ne sont pas en rapport avec le jeu : la monétisation, le tracking de données, les publicités.

Les projets précédents avaient été pensés uniquement pour iOS et avaient donc été développés avec Cocos 2D.

2.4. Comment avez-vous fait votre choix parmi les moteurs de jeux ? Qu'est-ce qui vous a décidé à prendre Unity ?

Le choix a été assez simple : on avait besoin d'un moteur 3D, souple et multiplateforme. Sur le marché, au niveau des moteurs matures, il en existe deux : l'Unreal Engine et Unity. Les coûts pour utiliser Unity sont bien moindres et la communauté est plus vaste. On peut facilement récupérer des expansions et des ressources graphiques que nous ne retrouvons pas chez le concurrent. Toutefois, l'équipe de développement de Heliceum possède des compétences pour les deux moteurs.

2.5. Avez-vous entendu parler de Project Anarchy ?

Oui. Le problème est qu'il vient de sortir et n'est pas forcément totalement fini. Le risque de production en prenant un nouveau moteur est énorme. On n'est pas certain de bien le prendre en main et on n'est pas sûr d'avoir un module de monétisation totalement fait à coûts réduits.
Chez Heliceum, on souhaite s'appuyer sur un moteur éprouvé pour ne pas faire toute la partie assurance qualité que n'a pas forcément faite la personne qui vend le moteur.

2.6. D'un point de vue moins entreprise, que pensez-vous de Project Anarchy ? Le conseilleriez-vous aux amateurs ?

Même d'un point de vue amateur, ce qui me fait peur, c'est qu'il n'est pas assez mature. Pour Unity, si je voulais faire un projet amateur dessus, je sais que je pourrais récupérer beaucoup de choses en ligne. Il y a aussi une très forte communauté qui l'utilise et je ne conseille donc pas de commencer par du Project Anarchy, mais plutôt par Unity, ou même de l'UDK.

Ce n'est pas pour cela qu'il ne faut pas regarder. Havok est expérimenté, notamment sur l'IA et la physique, donc cela dépend vraiment du projet que l'on veut faire.

2.7. Quel est votre point de vue sur les nouveaux appareils mobiles, basés sur Firefox OS ou Tizen ?

D'un point de vue professionnel, c'est compliqué d'aller vers une plateforme lorsqu'il y a très peu de personnes dessus. On arrive très vite à avoir plus de coûts à porter sur une telle plateforme que de gains. Sur une plateforme mobile, les achats sont au niveau de deux euros par personne et donc, il faut vendre en masse pour pouvoir rentabiliser les applications.

2.8. Si Unity permet l'exportation vers l'une de ces nouvelles plateformes et cela gratuitement, est-ce que cela change votre vision des choses ?

Cela dépendra des types d'applications. Par exemple, sur « Sale Gosse » qui est un jeu « Free To Play », pour qu'il marche, on est obligé de mettre plein d'autres choses que le jeu en lui-même. Comme je le disais, on a une grosse couche de monétisation qui reste dedans et qui fait appel à pas mal de prestataires externes. Si ceux-ci sont gérés dans un plugin Unity, on va pouvoir porter simplement sinon cela demandera énormément de travail. Le fait que Unity aille vers les plateformes, c'est le minimum. Il faut aussi que la communauté se dirige vers cette plateforme.
De plus, une nouvelle version de l'application demande beaucoup de coûts au niveau de l'assurance qualité afin de tester tous les matériels.

2.9. En effet, déjà qu'il y avait des différences entre les mobiles Apple et Android…

Tout à fait, maintenant, pour tester une version Apple, il y a beaucoup de versions à tester : iOS 5, 6 et 7, mais aussi pour les différentes résolutions d'écran, que ce soit téléphones (3.5", 4") ou tablettes, avec écran Retina ou non.

2.10. Pouvez-vous me donner une estimation du temps de développement pour « Sale Gosse » ?

C'est assez compliqué. Il y a eu beaucoup de travail sur le « look and feel » (le ressenti). Cela a entrainé beaucoup d'allers-retours entre les tests et l'équilibrage du jeu. En développement pur, on peut finir en trois mois lorsque les spécifications sont parfaites et que l'on n'a plus qu'à les rouler. Finalement, le jeu a pris plus de temps : environ six

mois. Ce n'est pas spécialement un projet complexe, mais nous apportons un grand soin à la finalisation et au ressenti du jeu.
Finalement, les tests de jouabilité ont duré trois mois et les vérifications de la qualité, environ un mois.

Trailer de Sale Gosse : Lien 74

2.11. Comment gérez-vous les allers-retours entre les développeurs, les testeurs et les game designers ?

Comme le studio est assez petit, la communication entre les membres des équipes est directe. Nous utilisons la méthode AGILE permettant de voir où on en est et ce qui a changé, au jour le jour.

2.12. Comment assurez-vous la qualité de vos jeux ?

On a un responsable qualité recruté pour cela. À chaque version délivrée, le jeu passe une ou deux journées à tester les non-régressions et à le malmener. Ensuite, il va créer un rapport de test et créer des tests dans notre base de bogues JIRA (Lien 75) (permettant aussi la gestion de projets). Ensuite les bogues sont répartis parmi les développeurs. La version corrigée revient au responsable devant la tester à nouveau et le processus continue jusqu'à avoir un nombre de bogues très limité (ou non bloquants pour la mise en production).

2.13. Y a-t-il des moyens d'automatiser les tests sur Android ou iOS ?

Dans le jeu vidéo, c'est très compliqué. Dans les autres domaines, on peut tester les programmes par des serveurs automatisés, car leurs sorties sont déterministes. On peut utiliser des tests unitaires, ou des tests de séquences. Dans le jeu vidéo, il y a une dépendance forte avec l'utilisateur et le hasard fait aussi partie de l'équation. Finalement, seuls les tests unitaires peuvent être utilisés et le reste sera testé durant la phase d'assurance qualité.

2.14. Avez-vous des anecdotes sur « Sale Gosse » ?

Lors du prototypage des lance-pierres, en voulant accélérer le rythme de tir, au lieu de tirer à chaque fois qu'on lâchait le doigt, à partir du moment où le doigt était lâché, à chaque image, une pierre partait. Cela faisait qu'il y avait des milliers ou millions de pierres qui volaient partout.

2.15. Pouvez-vous nous donner une estimation de la répartition des ressources dans le binaire de « Sale Gosse » ?

De façon générale, dans « Sale Gosse », ce sont les textures qui prennent le plus de place. Les modèles 3D sont limités notamment par les textures prenant environ 90 % de la place, les sons 5 % et les modèles 3D et le code, un peu moins.

2.16. Comment faites-vous pour optimiser la taille du binaire ?

Au début du projet, on a pris chacune des ressources de façon unitaire, on a regardé son poids et on a fait une projection par rapport au moment où on voulait les mettre dans le jeu permettant de savoir si on allait les voir dès le début et savoir comment limiter chacun de ses paramètres. De plus, au cours du projet on vérifiait si on respectait les différentes contraintes.

2.17. Comment réagissez-vous lorsqu'une texture prend trop de place ?

On parle au graphiste qui a mis la texture pour voir comment on peut la réduire :

- soit en réduisant la taille de la texture ;
- soit en réduisant la taille de l'atlas en enlevant certains éléments et en les mettant à l'extérieur (les atlas sont utilisés pour économiser la taille mémoire, faisant que l'on doit gérer un compromis entre taille en mémoire et taille du binaire).

Après, en réduisant les textures, la qualité, savoir si on a besoin d'autant de couleurs alors qu'elles ne sont pas forcément vues. Finalement, c'est un compromis entre la taille du binaire et ce qui est important au visuel. Si une texture est coûteuse et qu'elle est peu vue, alors il faut la compresser.

3. Outils dans le développement de jeux vidéo

3.1. Que pensez-vous des logiciels libres ?

Dans le monde des jeux vidéo, les logiciels GNU sont intensément utilisés notamment pour le scripting. Étant plus jeune, je me rappelle avoir envoyé des « patch notes » pour des logiciels GNU, par exemple Flex/Bison. Ce sont des choses sur lesquelles il faut continuer à travailler, qui aident tout le monde à avancer.

3.2. Quelle en est votre utilisation pour « Sale Gosse » ?

Les graphistes utilisent Blender pour les modèles 3D. De façon générale, il y a toujours un GIMP installé sur les machines des développeurs permettant d'éviter de payer une licence Photoshop pour les quelques essais. On a aussi des shells et des scripts permettant d'automatiser beaucoup de tâches. Un bon développeur est un développeur fainéant.

4. Modèle économique

4.1. Comment effectuez-vous le choix entre le « Free to play » et l'achat du jeu complet ?

Cela se fait selon le gameplay du jeu. On ne vise pas du tout les mêmes gens entre les deux modèles économiques. Typiquement, un jeu de plateau dans lequel on va faire les parties les unes après les autres sera payant. Un jeu sur lequel on peut avoir une première expérience qui est plaisante, sur lequel on pourra avancer assez loin et que l'intérêt du jeu va être à long terme sera « Free to Play ». On pourra rajouter des fonctionnalités dans le jeu et on demandera au joueur de payer au moment où il souhaite vraiment s'investir dans le jeu.
C'est vraiment une question de gameplay, il faut vraiment choisir le concept avant de déterminer le meilleur modèle économique lui correspondant.

5. Pour finir

5.1. Quels conseils donneriez-vous aux membres développeurs de jeux vidéo de Developpez.com ?

Bien viser la plateforme dans laquelle on a le plus de facilité. Souvent, quand on développe on a certaines affinités avec certains langages. Personnellement, je viens d'un endroit où c'était de l'informatique pure et dure et où j'ai appris le C++, donc de gérer sa mémoire à la main et être rigoureux. Mon premier projet amateur a été un jeu sur Nintendo DS, car c'était la plateforme la plus simple d'entrée en C++, mais cela demandait de la programmation bas niveau.
Si on arrive à faire plus facilement du script ou autre, on peut passer sur de l'UDK où il n'y aura quasiment besoin d'aucune ligne de code pour faire son premier projet. Seul un éditeur visuel suffit, dans lequel vous reliez des boites permettant ainsi d'avoir une première vision du développement en ayant la logique algorithmique. Ensuite, partir sur l'Unreal Script pour apprendre à structurer un programme.
Je conseille vraiment pour des gens qui n'ont pas vraiment développé d'aller sur les moteurs qui sont déjà assez utilisés tels Unity ou Unreal. Ce choix repose sur la grande communauté et le nombre de ressources sur lesquelles s'appuyer sans avoir à tout comprendre, notamment pour les subtilités qui ne servent qu'à l'optimisation ou les choses très avancées.

5.2. Vous ne parlez pas du tout du CryEngine ?

Il est moins grand public et si on veut faire du développement amateur autant utiliser des plateformes assez ouvertes. Le CryEngine est utilisé dans certaines écoles, mais si on commence à partir de rien, autant aller sur des endroits où il y a d'autres gens qui peuvent nous aider. Si on fait un projet qui a un minimum d'envergure et que l'on ne va pas pouvoir travailler tout seul, sous-entendu, créer une équipe, autant aller sur une solution où pas mal de gens ont des compétences.

Retrouvez l'article d'Alexandre Laurent en ligne : Lien 76

Spring Framework 4.0 est disponible, une nouvelle version majeure 4 ans après !

Après une première *Release Candidate* disponible depuis début novembre 2013 (Lien 77) et une seconde début décembre (Lien 78), le site officiel nous annonce la disponibilité de la version finale depuis le 12 décembre 2013.

Là où la version actuelle (3.2) basait ses principes sur un support Java SE 7 et Servlet 3.0, la nouvelle version présente de nombreuses nouveautés. On peut citer notamment :

- support de plusieurs fonctionnalités de base de Java SE 8 (lambda expressions, support de la nouvelle API Date-Time ...) et de Java EE 7 (Servlet 3.1, JPA 2.1 ...) ;
- compatibilité avec Groovy 2 (possibilité d'écrire des beans Groovy simplifiant la syntaxe ...) ;
- API Java pour les WebSockets ;
- mise à jour des dépendances pour améliorer la compatibilité de Spring avec des versions Java SE 6 ou plus.

En parallèle de cette nouvelle, il est aussi annoncé qu'une version Spring 4.0 compatible avec OpenJDK 8 est prévue au plus tôt pour le mois d'avril 2014.

Pour plus de détails, n'hésitez pas à faire un tour sur :

- Le site officiel : Lien 79
- Qu'est-ce qui est nouveau dans Spring Framework 4 ? Lien 80
- La page de téléchargement : Lien 81

Que pensez-vous des nouveautés de cette nouvelle version ? Vous semble-t-elle prometteuse ?

Commentez la news de mlny84 en ligne : Lien 82

JavaFX Scene Builder devient open source, dans le cadre du projet OpenJFX, pour Android et iOS les kits de développement sont également déjà publiés en open source

Sur la liste de diffusion du projet OpenJFX, Simon Vienot, directeur technique chez Oracle, vient d'annoncer le passage de JavaFX Scene Builder à l'open source. Il se sent satisfait de cette phase, en annonçant que l'ensemble des fonctionnalités de Scene Builder sont disponibles, y compris le Kit de l'API de Scene Builder (qui permet son intégration aux IDE) et l'application standalone Scene Builder.

À noter que la dernière version en date de JavaFX Scene Builder est la version 2.0 (Lien 83), mais il s'agit une version bêta. La version stable actuelle est la version 1.1 (Lien 84).

Pour rappel, JavaFX est une technologie composée d'un ensemble de bibliothèques Java (depuis la version 2.0 de JavaFX), permettant la création d'application « client riche » (RIA). JavaFX est multiplateforme, fonctionnant sous Windows, Linux et Mac OS. Il se veut le successeur de Swing pour les applications standalone bureau, mais fonctionne aussi sur le web sous forme d'Applet, sur mobile et sur l'embarqué, notamment sur Raspberry Pi. Oracle a déjà rendu open source les kits de développement en JavaFX pour Android et iOS .

En effet, sur iOS, il est possible de développer des applications JavaFX avec OpenJFX via la machine virtuelle RoboVM, qui permet de créer des applications Java natives pour iOS. Retrouvez un bon tutoriel pour la création avec Eclipse (Lien 85) et apprenez comment packager avec ce tutoriel (Lien 86) utilisant iPack. À noter que RoboVM n'est pas la seule alternative.

De plus, pour Android, Oracle a aussi publié en open source les runtime (Lien 87) pour le développement d'applications Android en JavaFX sous Linux et Mac OS, et la communauté a commencé à y travailler (Lien 88) pour améliorer la stabilité et la maturité de la plateforme. Vous pouvez également commencer à développer (Lien 89) et retrouver une démo publiée sur ce billet (Lien 90) d'une application JavaFX tournant sous Android et manipulant le multipoint et les gestures.

JavaFXAndroid Multitouch & gestures - YouTube : Lien 91

JavaFX Scene Builder est, quant à lui, un outil de création et de design d'interface graphique JavaFX, en glissant et déposant, modifiant les propriétés et personnalisant les feuilles de style CSS.

Simon Vienot annonce également que la seule partie qui reste encore fermée est le code source pour le paquetage natif et l'installeur.

Nous soulignons de même que la version bêta de JavaFX 8 est déjà disponible (Lien 92) et est une partie intégrante de JavaSE 8. Il apporte avec lui toutes les fonctionnalités de Java 8 notamment les expressions lambda et autres, et il ajoute le support 3D et plusieurs autres fonctionnalités.

Le code source du JavaFX Scene Builder 2.0 : Lien 93
Si vous voulez contribuer, inscrivez-vous au mailing-list du projet : Lien 94

Commentez la news de Bilal Soidik en ligne : Lien 95

Ceylon se lance à la conquête des développeurs Java, le langage de Red Hat basé sur les points forts de Java sort en version 1.0.0

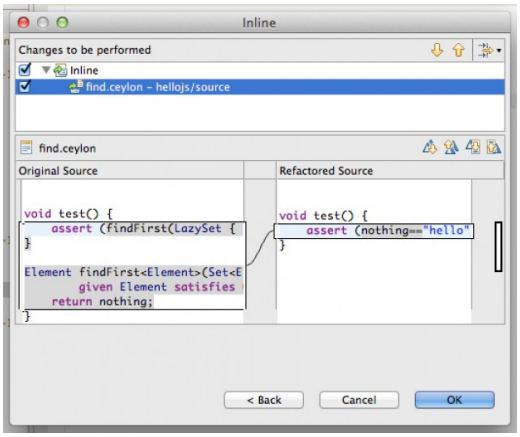
Les développeurs rebutés par certaines faiblesses de Java, dues entre autres à son âge avancé (environ 17 ans), ont désormais à leur disposition une alternative fondée sur ce qui a fait le succès de ce langage.

Après plus de trois ans de développement, Red Hat vient de dévoiler la version stable de Ceylon 1.0.0, son langage de programmation open source, conçu pour concurrencer directement Java.

Initié par Gavin King de Red Hat, connu dans le monde Java comme étant l'auteur d'Hibernate et du framework Seam, Ceylon se distance de Java en proposant des bibliothèques de classes plus modernes et une syntaxe favorable à la définition d'interface utilisateur.

Pour son auteur, Ceylon est un langage orienté objet facilement compréhensible avec un typage statique. Parmi les atouts-clés de Ceylon, on note :

- un accent mis sur la lisibilité et un penchant pour l'omission et l'élimination des parties potentiellement nocives ;
- un système de type extrêmement puissant, qui combine sous-type et polymorphisme paramétrique ;
- un traitement unique des fonctions et des types tuple, permettant une abstraction puissante ;
- une syntaxe flexible ;
- des types génériques entièrement réifiés, à la fois pour la JVM et les machines virtuelles JavaScript.



Ceylon 1.0.0 dispose d'un compilateur qui peut produire soit du bytecode Java ou JavaScript. Ce qui suppose que le même code peut s'exécuter sur une JVM ou sur un environnement d'exécution JavaScript comme Node.js.

Ceylon 1.0.0 est accompagné de son SDK, qui dispose de nouveaux modules ceylon.build (un framework pour l'écriture des scripts de compilation en Ceylon) et ceylon.html (une bibliothèque pour définir du contenu HTML en Ceylon).

La version 1.0.0 de l'environnement de développement de Ceylon est également disponible. Basé sur Eclipse, l'EDI supporte l'autocomplétion, le refactoring, la compilation incrémentale, etc. Certaines fonctionnalités ont été améliorées comme la vue hiérarchique des types, la coloration syntaxique et la recherche.

Le code source de Ceylon est disponible en open source sur GitHub.

Red Hat planche déjà sur la version 1.1, dont la feuille de route a été divulguée. L'attention de la société portera essentiellement sur des améliorations des performances du langage, de son compilateur et de son SDK.

Les outils liés au langage : Lien 96
Télécharger le SDK de Ceylon : Lien 97
Télécharger le code source de Ceylon sur GitHub : Lien 98
Télécharger l'EDI de Ceylon : Lien 99

Commentez la news de Hinault Romaric en ligne : Lien 100

NetBeans 7.4 : support du JDK 8, nouvelles fonctionnalités HTML5, optimisation des performances et développement mobile pour Android et iOS, l'EDI open source sort

NetBeans, l'environnent de développement polyglotte open source franchit un nouveau cap. Oracle vient de publier la version finale de NetBeans 7.4, qui offre un support amélioré du développement mobile.

Cette version apporte comme nouveautés phares :

- le développement HTML5 pour la création d'applications Android et iOS ;
- la prise en charge du HTML5 pour les développeurs Java EE et PHP ;
- le support de la prochaine version majeure de Java (Java SE 8) ;
- une réimplémentation de JavaFX selon l'architecture du JDK 8 ;
- le support de PhoneGap (Framework pour le développement d'applications mobiles Web hybrides) ;
- et plusieurs autres améliorations, optimisations et nouveautés pour l'éditeur de code.

Présentation de NetBeans 7.4 : Lien 101

Télécharger NetBeans 7.4 : Lien 102

Le Wiki du projet : Lien 103

Commentez la news de Hinault Romaric en ligne : Lien 104

Liens

Lien 01 : http://www.developpez.com/actu/63029/
Lien 02 : https://www.apple.com/fr/ipad-air/
Lien 03 : http://www.apple.com/fr/ipad-mini/
Lien 04 : http://www.developpez.net/forums/d1387833/systemes/mac/keynote-apple-22-10-nouveaux-ipad-air-ipad-mini-retina/
Lien 05 : https://www.apple.com/fr/osx/
Lien 06 : http://www.developpez.net/forums/d1387812/systemes/mac/keynote-apple-22-10-os-x-mavericks-ilife-iwork-disponibles-gratuitement/
Lien 07 : https://www.apple.com/fr/macbook-pro/
Lien 08 : https://www.apple.com/fr/mac-pro/
Lien 09 : http://www.developpez.net/forums/d1387823/systemes/mac/keynote-apple-22-10-nouveau-macbook-pro-nouveau-mac-pro/
Lien 10 : http://developer.apple.com/library/ios/#documentation/MacOSX/Conceptual/BPInternational/Articles/LanguageDesignations.html
Lien 11 : https://dl.dropboxusercontent.com/u/9955727/Applications/Localisation.zip
Lien 12 : http://seelass.developpez.com/tutoriels/devAppiOSmultilingue/
Lien 13 : http://perl.developpez.com/livres/?page=LivresDivers#L9781558607019
Lien 14 : http://laurent-rosenfeld.developpez.com/tutoriels/perl/programmation-fonctionnelle/operateurs-listes/
Lien 15 : http://dico.developpez.com/html/1710-Internet-Ajax-Javascript-Asynchrone-et-XML.php
Lien 16 : http://dico.developpez.com/html/1696-Langages-HTML-HyperText-Markup-Language.php
Lien 17 : http://dico.developpez.com/html/1693-Internet-CSS-Cascading-Style-Sheets.php
Lien 18 : http://dico.developpez.com/html/278-Langages-PHP-PHP--Hypertext-Preprocessor.php
Lien 19 : http://dico.developpez.com/html/155-Langages-J2EE-Java-2-Entreprise-Edition.php
Lien 20 : http://dico.developpez.com/html/273-Langages-ASPNET-Active-Server-Pages-NET.php
Lien 21 : http://dico.developpez.com/html/37-Internet-XML-eXtended-Markup-Langage.php
Lien 22 : http://dico.developpez.com/html/910-Langages-JSP-Java-Server-Pages.php
Lien 23 : https://github.com/medikoo/domjs
Lien 24 : http://jade-lang.com/
Lien 25 : http://ejohn.org/blog/javascript-micro-templating/
Lien 26 : http://embeddedjs.com/
Lien 27 : http://olado.github.io/doT/
Lien 28 : https://github.com/janl/mustache.js/
Lien 29 : http://handlebarsjs.com/
Lien 30 : http://akdubya.github.io/dustjs/
Lien 31 : http://twitter.github.io/hogan.js/
Lien 32 : http://beebole.com/pure/
Lien 33 : http://leonidas.github.io/transparency/
Lien 34 : http://garann.github.io/template-chooser/
Lien 35 : http://sylvainpv.developpez.com/tutoriels/javascript/guide-templating-client/
Lien 36 : http://imikado.developpez.com/tutoriels/php/presentation-mkframework/
Lien 37 : http://localhost/mkframework
Lien 38 : http://imikado.developpez.com/tutoriels/php/creer-votre-microblogging/
Lien 39 : http://www.html5rocks.com/en/tutorials/webcomponents/customelements/
Lien 40 : http://www.whatwg.org/specs/web-apps/current-work/multipage/
Lien 41 : https://dvcs.w3.org/hg/webcomponents/raw-file/tip/spec/custom/index.html
Lien 42 : https://dvcs.w3.org/hg/webcomponents/raw-file/tip/explainer/index.html
Lien 43 : http://www.whatwg.org/specs/web-apps/current-work/multipage/elements.html#htmlunknownelement
Lien 44 : https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object/defineProperty
Lien 45 : https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/get
Lien 46 : http://www.html5rocks.com/tutorials/webcomponents/shadowdom/
Lien 47 : http://www.html5rocks.com/tutorials/webcomponents/shadowdom-201/
Lien 48 : https://dvcs.w3.org/hg/webcomponents/raw-file/tip/spec/templates/index.html
Lien 49 : http://www.html5rocks.com/tutorials/webcomponents/template/
Lien 50 : http://www.polymer-project.org/articles/styling-elements.html
Lien 51 : http://www.html5rocks.com/tutorials/webcomponents/shadowdom-201/
Lien 52 : http://www.polymer-project.org/articles/styling-elements.html#preventing-fouc
Lien 53 : http://polymer-project.org/
Lien 54 : http://www.polymer-project.org/platform/custom-elements.html
Lien 55 : http://www.x-tags.org/
Lien 56 : http://lists.w3.org/Archives/Public/public-webapps/2013JulSep/0287.html
Lien 57 : https://dvcs.w3.org/hg/webcomponents/raw-file/tip/explainer/index.html
Lien 58 : http://www.html5rocks.com/en/
Lien 59 : http://dmouronval.developpez.com/tutoriels/html5/elements-personnalises/
Lien 60 : http://www.courtois.cc/murphy/murphy.html
Lien 61 : http://www.wi-fi.org/
Lien 62 : http://www.commentcamarche.net/
Lien 63 : http://caleca.developpez.com/tutoriels/technologie-wifi/
Lien 64 : http://opengl.developpez.com/docs/man/man/glOrtho
Lien 65 : http://fr.wikipedia.org/wiki/Distance_%28mathématiques%29
Lien 66 : http://jeux.developpez.com/tutoriels/theorie-des-collisions/formes-2d-simples/
Lien 67 : http://www.heliceum.com/
Lien 68 : http://www.dont-nod.com/
Lien 69 : http://www.heliceum.com/app/fakers/
Lien 70 : http://www.heliceum.com/app/human-defense/
Lien 71 : http://www.heliceum.com/app/sale-gosse/
Lien 72 : http://www.heliceum.com/app/le-grand-defi-figaro/
Lien 73 : http://www.youtube.com/v/7-o9ptPuQ74&autoplay=1
Lien 74 : http://www.youtube.com/v/tguXUj1wpVU&autoplay=1
Lien 75 : https://www.atlassian.com/software/jira
Lien 76 : http://jeux.developpez.com/interviews/heliceum/directeur-technique-sebastien-broussaud/
Lien 77 : https://spring.io/blog/2013/11/01/spring-framework-4-0-rc1-available

Lien 78 : https://spring.io/blog/2013/12/03/spring-framework-4-0-rc2-available
Lien 79 : http://spring.io/
Lien 80 : http://docs.spring.io/spring/docs/4.0.0.RELEASE/spring-framework-reference/htmlsingle/#spring-whats-new
Lien 81 : http://projects.spring.io/spring-framework/
Lien 82 : http://www.developpez.net/forums/d1400433/java/general-java/spring/spring-framework-4-0-disponible/
Lien 83 : http://www.oracle.com/technetwork/java/javafx/downloads/devpreview-1429449.html
Lien 84 : http://www.oracle.com/technetwork/java/javafx/downloads/index.html
Lien 85 : http://tomsondev.bestsolution.at/2013/11/06/develop-a-javafx-ios-app-with-robovm-efxclipse-tools-in-10-minutes/
Lien 86 : https://blogs.oracle.com/jfxprg/entry/ipack_the_ios_application_packager
Lien 87 : https://bitbucket.org/javafxports/android/downloads
Lien 88 : https://bitbucket.org/javafxports/android/wiki/Home
Lien 89 : https://blogs.oracle.com/jfxprg/entry/building_javafx_for_android_with
Lien 90 : https://blogs.oracle.com/jfxprg/entry/javafx_on_android_multitouch_and
Lien 91 : http://www.youtube.com/watch?v=qKmG3e9d5ks
Lien 92 : http://jdk8.java.net/
Lien 93 : http://hg.openjdk.java.net/openjfx/8/graphics/rt/rev/4e5dd280c928
Lien 94 : http://mail.openjdk.java.net/mailman/listinfo/openjfx-dev
Lien 95 : http://www.developpez.net/forums/d1398526/java/interfaces-graphiques-java/javafx/javafx-scene-builder-devient-open-source-cadre-projet-openjfx/
Lien 96 : http://ceylon-lang.org/documentation/1.0/reference/tool/ceylon/subcommands/index.html
Lien 97 : https://modules.ceylon-lang.org/categories/SDK
Lien 98 : https://github.com/ceylon
Lien 99 : http://ceylon-lang.org/documentation/1.0/ide/features/
Lien 100 : http://www.developpez.net/forums/d1393210/java/general-java/ceylon-se-lance-conquete-developpeurs-java/
Lien 101 : http://www.youtube.com/watch?v=KHtMn4E1ubA
Lien 102 : https://netbeans.org/downloads/
Lien 103 : http://wiki.netbeans.org/NewAndNoteworthyNB74
Lien 104 : http://www.developpez.net/forums/d1365938/java/edi-outils-java/netbeans/netbeans-7-4-support-jdk-8-nouvelles-fonctionnalites-html5-optimisation-performances/