Developpez

Le Mag

Édition de février - mars 2014.
Numéro 50.
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

C'est le retour du printemps, des beaux jours, les animaux se réveillent, tout comme les rédacteurs de developpez.com, les arbres fleurissent, les articles aussi.

Profitez-en bien !

La rédaction

Article Développement Web

Règles d'application des styles CSS et gestion des conflits

Cet article va vous expliquer les trois principes permettant d'appliquer des styles à un élément : la cascade, l'héritage et la spécificité.

par **Didier Mouronval**
Page 34

Article Delphi

Introduction aux Styles FireMonkey XE4 – Premiers pas

Pour les « vieux delphistes », l'éditeur de style est déroutant. Évitez les principaux pièges des styles FireMonkey.

par **Christian Caleca**
Page 34

Tile Mapping : Présentation générale

Dans cette partie, nous allons présenter le « Tile Mapping », et faire un petit programme qui affiche un petit monde simple.

1. Introduction

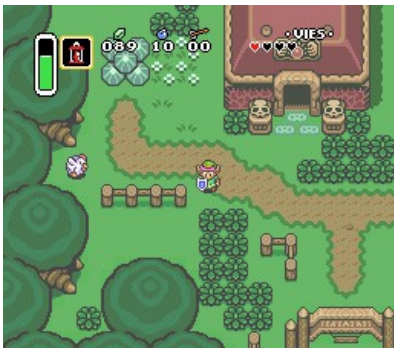
Bienvenue dans la première partie de ce tutoriel.
Dans cette partie, nous allons présenter le « Tile Mapping », et faire un petit programme qui affiche un petit monde simple de deux façons différentes :

- avec des nombres directement dans le code ;
- avec un fichier texte comme modèle.

2. Problématique

Vous avez déjà tous joué à ce qu'on appelle des jeux de plates-formes en 2D. Il s'agit de jeux où un personnage court dans un monde, parfois à toute vitesse, saute, monte sur des blocs, et infatigablement continue à courir et à sauter…
Pendant que vous courez, l'écran défile. Le monde que vous parcourez peut être plus ou moins grand.
Un exemple très connu de jeu de plates-formes, duquel nous allons nous inspirer, est Super Mario Bros.

Vous avez déjà sûrement joué aussi à des jeux vus de dessus, comme ce bon vieux Zelda.



Dans ce tutoriel, nous allons essayer de voir comment de tels jeux sont faits. Comment le monde est mis en place et comment faire défiler l'écran.
Comment, avec SDL, peut-on arriver à faire un tel type de jeu ? Comment avoir quelque chose de rapide et d'efficace ?

2.1. Cheminement du tutoriel

Ce tutoriel devrait grandir. Voici ce qu'il pourrait enseigner au fur et à mesure des versions.

2.1.1. Présentation des techniques

Les jeux de plates-formes ont très rapidement adopté la technique du « Tile-Mapping », depuis leur plus jeune âge. Je présenterai tout d'abord rapidement une technique pleine d'inconvénients, puis nous passerons sur la technique du tile mapping.

2.1.2. Création d'un mini monde avec des chiffres

Nous verrons comment créer un petit monde, d'un seul écran, qui ne bouge pas, grâce à un tableau de chiffres.

2.1.3. Mise en place de quelques propriétés, isolement du code

Pour un jeu de plates-formes, il sera important de définir où est le sol, où est le ciel, de façon à ce que par la suite, notre personnage puisse évoluer dans le monde logiquement.
Pour un jeu vu de dessus, il sera important de savoir où on a le droit de marcher et où on n'a pas le droit.
Les exemples fournis seront découpés en couches de façon à bien isoler la gestion du monde du reste, et qu'il soit facile, avec une seule fonction, d'afficher un niveau.

2.1.4. Scrolling

Nous verrons ensuite comment faire défiler l'écran (on parle de scrolling), c'est-à-dire comment faire bouger tout le décor de façon à ce que la caméra suive un personnage et que le fond défile derrière lui. Tout cela sera également très facile à manipuler au niveau du code.

2.1.5. Insertion d'un personnage

Nous verrons finalement en deuxième partie, comment insérer un personnage dans un décor, comment faire en sorte que la caméra le suive automatiquement et comment faire en sorte qu'un mur l'arrête (collisions).

2.1.6. Évolution

À l'heure où j'écris ces lignes, le plan futur de ce tutoriel n'est pas encore écrit, mais nous pourrons envisager les points suivants (en fonction de vos commentaires),
en vrac :

- des « tuiles » animées ;
- le scrolling en plusieurs couches (derrière, ça défile moins vite que devant) ;
- des objets qu'on pourrait ramasser ;

- des ennemis qu'on pourrait insérer ;
- plusieurs personnages, avec une caméra intelligente ;
- des blocs cassables ;
- des pentes, des échelles ;
- etc.

3. Technique de l'image figée

Avant de parler « Tile Mapping », voici une technique qui pourrait être utilisée pour faire un jeu de plates-formes.
L'idée qui vient à l'esprit tout de suite est de dessiner son monde sous un logiciel de dessin, « Paint » par exemple. L'idée serait de charger l'image au démarrage du programme, de l'afficher en tant que fond d'écran, puis afficher un Mario par-dessus. Les inconvénients sont les suivants.

3.1. C'est coûteux en mémoire

En effet, une grande image, c'est parfois plusieurs mégaoctets de mémoire. Si vous voulez faire un grand monde, multipliez par le nombre d'images nécessaires et vous obtiendrez une utilisation mémoire inacceptable…

3.2. C'est inexploitable

Le plus gros inconvénient est que c'est inexploitable. En effet, si vous affichez votre image de fond, puis que vous affichez Mario, pouvez-vous dire facilement si Mario est sur une plateforme ? Dans l'air ? Dans un mur ? Que s'il avance d'un pas, il tombe ?
Eh non… Vous pouvez éventuellement vous en sortir sur une image où le fond est uni, mais si vous avez une image de fond comme CastleVania ci-dessous, vous ne pouvez pas vous en sortir de cette façon…

Cette technique a trop d'inconvénients pour être utilisée. Je voulais en parler, car c'est souvent la première idée qui vient, de « dessiner » son monde, mais nous allons l'oublier, et passer enfin à la technique dont je vous parle depuis tout à l'heure…

4. Présentation du tile mapping

Avant de définir ce qu'est le Tile Mapping, nous allons ensemble regarder quelques images de jeux connus. Comme on dit « un bon croquis vaut mieux qu'un long discours, cela devrait nous aider.

Quelle est la particularité des cartes de ces jeux ?
Eh bien nous pouvons constater que des motifs se répètent. En effet, les sols sont des briques identiques, collées les unes à côté des autres.
Mieux que ça, on peut remarquer la régularité parfaite de la chose : les points d'interrogation de Mario sont exactement au-dessus des briques de sol.
Pareil, dans Zelda, que nous voyons en dessous, il y a des motifs identiques qui se répètent avec une grande régularité.
Le concept de « Tile Mapping » est de coller côte à côte des tuiles (« tiles » en anglais) dans une zone régulière. Pour cela, nous subdivisons l'écran en une grille régulière, et nous mettrons un carreau dans chaque case.
Vous voulez voir la grille ?

Si on regarde bien cette dernière image, et qu'on oublie Mario, l'ennemi, l'étoile, et les nuages, qui sont des sprites, nous avons affaire à un décor très régulièrement placé. Chaque brique s'emboîte parfaitement dans la grille.
Comme déjà dit, chacune de ces petites briques est appelée tuile, ou « tile ».
Il y a les tuiles uniques, comme les briques et les points d'interrogation, et les tuiles composées, comme le pot de fleurs, qui est plus gros qu'une case. Cependant, ce pot de fleurs, bien qu'il semble être un objet unique, sera vu par la machine comme huit cases infranchissables…
L'avantage d'une telle méthode est donc qu'au lieu de définir le monde (considérons qu'il ne bouge pas) par une grande image, on le définit par une grille de 13*15 cases.

5. Coût mémoire

Nous disions que nous définissons l'image d'au-dessus par 13*15 cases.
Cela fait 195 cases.
Combien y a-t-il de tuiles différentes ?
Sur notre image, on a :

- le bloc ciel ;
- le bloc sol ;
- le point d'interrogation ;
- quatre tuiles différentes pour le pot de fleurs.

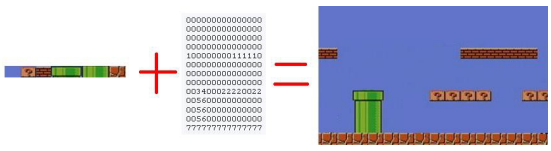
… moins d'une dizaine…
Nous pouvons imaginer un tableau de 13 * 15 cases qui contiennent un nombre. Si le nombre est 0, on met du « ciel », si le nombre est 1, on met un bloc cassable, si c'est 2, on met un « ? », 3 le bord supérieur gauche du pot, 4 le bord supérieur droit, 5 le bord gauche, 6 le bord droit, 7 le sol d'en bas, etc.
On définit, pour notre Mario, le tableau suivant :

```
000000000000000
000000000000000
000000000000000
000000000000000
100000000111110
000000000000000
000000000000000
000000000000000
003400022220022
005600000000000
005600000000000
005600000000000
777777777777777
```

Ce tableau de nombres décrit parfaitement notre monde, car il nous dit, pour chaque case, quel bloc mettre.
Vous suivez toujours ?
Du coup, avec :

- quelques tuiles,
- un tableau de nombres,

on définit un monde ! Schématiquement, cela donne :



La partie de gauche s'appelle « TileSet ». Elle contient les différents carreaux à poser. Ce sont les Tilesets qui définissent le graphisme. Dans mon cas, j'ai fait un tileset d'une seule ligne, mais comme les jeux contiennent quand même davantage de tiles, on définit souvent un tileset sur plusieurs lignes.
Vous trouverez de nombreux exemples sur Google Images en cherchant « tileset » !
La partie du milieu est le tableau de correspondance.
La partie de droite est bien sûr le résultat final.
Revenons maintenant au coût mémoire.
Pour faire mon monde de Mario, j'ai besoin d'un ensemble de tuiles (une petite image en soi), et du tableau.
Si je considère que je n'aurai pas plus de 256 tuiles différentes (je tape très large, et c'est souvent le cas), je peux compter un octet par case de mon tableau.
Avec mon tableau de 13 * 15, j'ai moins de 200 octets…. C'est très petit.
Maintenant, supposons un monde entier de Mario (qui défile). Imaginons le monde déplié ainsi :

Combien il y a-t-il de cases là-dedans ? À la louche je dirais 300 en largeur et 20 en hauteur.
300 * 20 = 6000 octets.
Voilà, le monde tout entier tient sur une petite image de tuiles + 6 Ko de données : une cacahuète quoi…
Et avec ça, on fait un grand monde.
C'est ainsi qu'ont procédé les consoles 8 et 16 bits, qui n'avaient pas beaucoup de mémoire.
Imaginez que si on avait codé tout le monde sous forme d'une graaaande image, on en aurait eu pour des dizaines de mégaoctets, pour une seule « texture », ce qui aurait bien chargé la carte graphique, et qui, outre ceci, aurait été inexploitable par la suite. (Nous verrons les avantages du « Tile Mapping » lorsque nous parlerons de collisions avec le décor.)

6. Code exemple

L'algorithme n'est pas complexe. Nous définissons, une fois pour toutes, une longueur et une hauteur de tuiles (qui resteront fixes).
Puis nous faisons un double for (i,j) sur le tableau, et nous affichons la bonne tuile à la position (i * largeur, j * hauteur).
Voici l'exemple suivant en C qui reconstruit le petit monde de Mario (juste la partie qu'on a étudiée)
Téléchargez et dézippez l'ensemble des fichiers de ce tutoriel ci-dessous :

Sources : Lien 01

6.1. Explication sur les programmes

Vous pouvez constater que le fichier téléchargé contient plusieurs programmes. Je vous dirai au fur et à mesure du tutoriel quel programme ouvrir.
Chaque programme a été fait avec Visual C++ 2008 Express, mais pourra être compilé avec d'autres versions, avec Code::Blocks, GCC, etc.
Si vous ouvrez le répertoire prog1, en dessous, vous avez un autre répertoire prog1, ainsi qu'un fichier .sln. Si vous avez Visual C++, double-cliquez sur le sln : le projet s'ouvre et est prêt à compiler.
Sinon, ouvrez le sous-répertoire prog1. Vous voyez un .vcproj (également pour Visual C++), et les sources et images utilisées.
Tous les projets sont faits de la même façon.
Ouvrez maintenant le projet prog1.
Vous pouvez le compiler et le lancer, ça doit marcher tout seul. (Je rappelle que vous devez avoir de bonnes bases en C,et avec la bibliothèque SDL pour poursuivre ce tutoriel.)

prog1.c

```
#include <SDL/SDL.h>


#pragma comment (lib,"sdl.lib")      // ignorez
ces lignes si vous ne linkez pas les lib de cette
façon.
#pragma comment (lib,"sdlmain.lib")
```

```
#define LARGEUR_TILE 24  // hauteur et largeur des tiles.
#define HAUTEUR_TILE 16

#define NOMBRE_BLOCS_LARGEUR 15  // nombre a afficher en x et y
#define NOMBRE_BLOCS_HAUTEUR 13

char* table[] = {
"000000000000000",
"000000000000000",
"000000000000000",
"000000000000000",
"100000000111110",
"000000000000000",
"000000000000000",
"000000000000000",
"003400022220022",
"005600000000000",
"005600000000000",
"005600000000000",
"777777777777777"};


void Afficher(SDL_Surface* screen,SDL_Surface* tileset,char** table,int nombre_blocs_largeur,int nombre_blocs_hauteur)
{
    int i,j;
    SDL_Rect Rect_dest;
    SDL_Rect Rect_source;
    Rect_source.w = LARGEUR_TILE;
    Rect_source.h = HAUTEUR_TILE;
    for(i=0;i<nombre_blocs_largeur;i++)
    {
        for(j=0;j<nombre_blocs_hauteur;j++)
        {
            Rect_dest.x = i*LARGEUR_TILE;
            Rect_dest.y = j*HAUTEUR_TILE;
            Rect_source.x = (table[j][i]-'0')*LARGEUR_TILE;
            Rect_source.y = 0;

SDL_BlitSurface(tileset,&Rect_source,screen,&Rect_dest);
        }
    }
    SDL_Flip(screen);
}

int main(int argc,char** argv)
{
    SDL_Surface* screen,*tileset;
    SDL_Event event;
    SDL_Init(SDL_INIT_VIDEO);        // prepare SDL
    screen = SDL_SetVideoMode(LARGEUR_TILE*NOMBRE_BLOCS_LARGEUR, HAUTEUR_TILE*NOMBRE_BLOCS_HAUTEUR, 32,SDL_HWSURFACE|SDL_DOUBLEBUF);
    tileset = SDL_LoadBMP("tileset1.bmp");
    if (!tileset)
    {
        printf("Echec de chargement tileset1.bmp\n");
        SDL_Quit();
        system("pause");
        exit(-1);
    }
```

```
Afficher(screen,tileset,table,NOMBRE_BLOCS_LARGEUR,NOMBRE_BLOCS_HAUTEUR);

    do  // attend qu'on appuie sur une touche.
    {
        SDL_WaitEvent(&event);
    } while (event.type!=SDL_KEYDOWN);

    SDL_FreeSurface(tileset);
    SDL_Quit();
    return 0;
}
```

Le code n'est pas complexe :
je veux afficher une image de 15 * 13 tuiles (NOMBRE_BLOCS_LARGEUR et NOMBRE_BLOCS_HAUTEUR dans les #define).
Chaque tuile fait 24 * 16 pixels (définis dans LARGEUR_TILE et HAUTEUR_TILE).
Dans le main, j'initialise SDL, la taille de l'image finale, obtenue en faisant l'opération nombre de cases en X * taille d'une tuile, et pareil pour y, bien entendu.
Je charge l'image des tuiles et je lance la fonction Afficher.
J'attends qu'on appuie sur une touche pour quitter.
Dans la fonction afficher, je définis deux SDL_Rect, celui de destination dont vous avez l'habitude, et celui source qui sera passé en second paramètre de SDL_BlitSurface pour un blit partiel.
Je fixe Rect_source.w et .h une fois pour toutes, car les tuiles auront toujours la même largeur et la même hauteur.
Ensuite, je fais un double for. Je fixe Rect_dest.x et y à la bonne position (qui dépend de i et de j), puis je définis Rect_source.x, qui lui dépend directement du nombre correspondant. (Nous allons détailler cette ligne ci-dessous.)
Rect_source.y est lui toujours à 0, car dans mon image des tuiles, toutes les tuiles partent de y=0.
Détaillons la ligne suivante :

```
Rect_source.x = (table[j][i]-'0')*LARGEUR_TILE;
```

Nous avons le tableau table défini, en dur, en haut du code.
Nous voulons récupérer le chiffre à la colonne i, ligne j.
D'où l'idée de faire table[i][j].
Cependant, vous remarquerez que j'ai mis [j][i] et non [i][j]. Cela vient de la définition même du tableau dans le code.
Prenons l'exemple avec des mots plutôt que des chiffres :

```
char* table[] = {
"Bonjour",
"Salut!!",
"Hello!!",
"Saloute",
"Hola!!!"
}
```

Si vous choisissez table[1], vous avez « Salut », puis table[1][2] pour avoir le 'l' de Salut.
Donc la lecture d'un tableau de chaîne se fait en ligne/colonne, alors que nous attendons colonne/ligne dans un repère 2D, d'où la transposée [j][i] au lieu de [i][j].
Maintenant, deuxième souci, pourquoi est-ce que je fais -'0' ?
Si je prends table[j][i], je ne tombe pas sur un nombre,

mais sur un caractère, sur le code ASCII de '1', celui de '0' etc.
Or le code ASCII de '0' vaut 48. Moi, je ne veux pas 48, je veux 0. En soustrayant '0' à un chiffre en ASCII, on obtient le chiffre réel. C'est une astuce classique, sachant que les chiffres sont contigus dans la table ASCII.
Et voilà, nous avons notre monde de Mario, à partir d'un ensemble de tuiles et de la table de correspondance. Heureux ?

(J'ai peut-être inversé deux tuiles dans mon dessin, mais bon…)
Respirez, et repensez à ce concept à tête reposée. L'essentiel est surtout de comprendre l'idée. Le code va peu à peu évoluer, et surtout se retrouver enfermé dans un autre fichier de façon à vous fournir des fonctions puissantes et simples à utiliser.

Retrouvez l'article de Fvirtman en ligne : Lien 02

Développement de jeux avec MonoGame - Partie I

Cet article est le premier d'une série qui va traiter du développement de jeux vidéo en C# avec la bibliothèque MonoGame.

C'est en même temps un recueil de mon expérience vu que j'étudie cette bibliothèque tout en écrivant ces articles, donc soyez indulgents quant à la qualité des codes et des informations que je vous transmets dans ces différents tutoriels.

1. Qu'est-ce que MonoGame ?

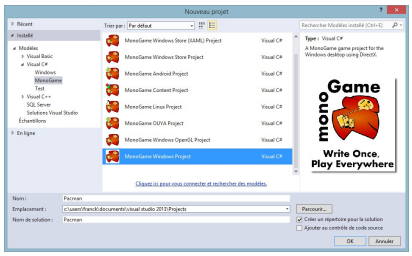
MonoGame est une implémentation Open Source du framework Microsoft XNA 4. Le but de cette implémentation est d'offrir la possibilité aux développeurs Xbox 360, Windows et Windows Phone, de porter leurs jeux sur iOS, Android, Mac OS X, Linux et Windows store. D'autres plates-formes seront supportées plus tard comme PlayStation.
Site officiel : Lien 03

2. Prérequis

Vu que nous allons utiliser C# autant vous dire (si ce n'est déjà fait) d'installer Visual Studio Express 2013 pour Windows Desktop (version utilisée pour cette série de tutoriels) : Lien 04
Il vous faudra aussi… MonoGame, bien sûr ! Je vous donne ci-dessous le lien de téléchargement de la dernière version empaquetée pour Windows qui prend en charge les templates pour VS2013 et antérieurs, il vous suffit de cliquer sur le lien dans la page : Lien 05
Il vous faudra également de bonnes bases en C#, si ce n'est pas le cas, lisez l'article Introduction au langage C# (Lien 06).
Nous pouvons commencer… Je sens que vous avez hâte !

3. Création de notre projet

Lancez donc Visual Studio et créez un nouveau projet C# MonoGame de type **MonoGame Windows Project** et nommez-le « Pacman »… Non, nous n'allons pas faire le jeu en lui-même, mais cela sera notre terrain d'entraînement, car c'est un jeu relativement simple et facile d'accès. Tous les tutoriels de cette série seront donc basés sur ce thème.



Une fois validé, Visual Studio vous génère le squelette du projet (encore heureux). Vous disposez d'une première classe nommée **Game1** (Game1.cs), renommez-la en **MyPacman.cs** que ce soit un peu plus attrayant et explicite ! Au message, cliquez sur **Oui** et tout le code sera adapté à ce changement ! On dispose donc d'une classe prête à l'emploi avec son constructeur par défaut ainsi que les méthodes suivantes :

- **Initialize :** sert à l'initialisation de variables et autres paramètres ;
- **LoadContent :** c'est ici par exemple que nous allons charger les images ;
- **UnloadContent :** partie dédiée au déchargement des images et autres ressources non managées par la plate-forme .Net donc, les ressources qui ne se trouvent pas dans le dossier **Content** du projet. Nous verrons l'utilité de ce dossier plus loin ;
- **Update :** ici viendra tout le code pour mettre à jour l'état du monde avec les nouvelles coordonnées par exemple de notre personnage et de ses ennemis et gérer les collisions entre autres ;
- **Draw :** nous appellerons ici nos fonctions de dessin à proprement parler donc l'affichage aux nouvelles coordonnées de nos images, du moins pour celles qui sont en mouvement.

Avec ceci nous avons également le fichier **Program.cs** (comme d'habitude en C#) où se trouve le point d'entrée du programme et dans lequel se trouve déjà la création d'une instance de notre classe **MyPacman** ! Vous pouvez essayer de lancer le programme, vous obtiendrez normalement une fenêtre avec un fond bleu !

4. Quelques préparatifs

Avant toute chose, il convient de préparer le terrain. Nous avons besoin d'une taille pour notre fenêtre, adaptée par rapport à l'image de fond qui représentera le monde dans lequel notre héros évoluera. Il nous faut aussi des classes qui représenteront les différentes parties du jeu, scène, héros, monstres. Pour cela, nous allons créer différentes classes.

4.1. La taille de la fenêtre

Je vous préviens tout de suite, ici les sprites (images) sont minuscules, car je les ai récupérés sur le site The Sprites Ressources (Lien 07). Ce site propose des sprites d'origine des anciens jeux essentiellement issus des consoles comme GameBoy, SNES, etc.
Donc, dans notre classe **MyPacman**, après les déclarations suivantes :

```
GraphicsDeviceManager graphics;
SpriteBatch spriteBatch;
```

Insérez ces lignes qui sont des constantes qui vont nous permettre de définir la taille de notre fenêtre :

```
public const int WINDOW_WIDTH = 224;
public const int WINDOW_HEIGHT = 248;
```

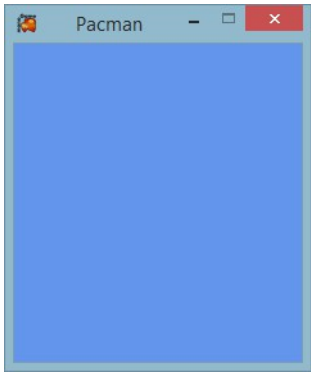
Ensuite, dans le constructeur, après les initialisations de base, ajoutez ces deux lignes :

```
graphics.PreferredBackBufferWidth = WINDOW_WIDTH;
graphics.PreferredBackBufferHeight =
WINDOW_HEIGHT;
```

Ajoutez ces mêmes lignes dans la méthode **Update** juste après le commentaire. Les commentaires de type :

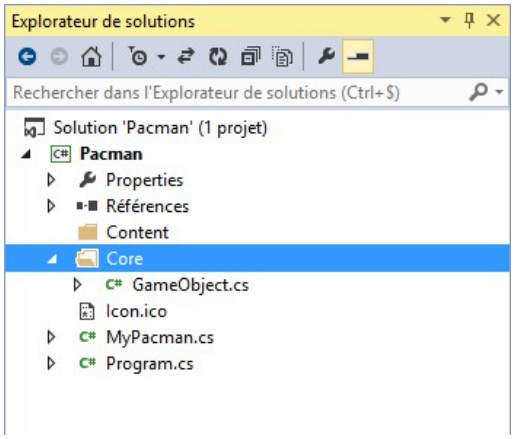
```
// TODO: Add your update logic here
```

vous montrent l'endroit où vous devez/pouvez ajouter votre code. La fenêtre devrait maintenant ressembler à ceci :



4.2. La classe de base de nos objets

Il nous faut une classe mère pour nos objets. Tous les objets ont les mêmes propriétés de base à savoir la taille, la position ainsi que la faculté de se dessiner eux-mêmes. Faites un clic droit sur le nom du projet dans l'explorateur de solutions puis, ajoutez un nouveau dossier que nous appellerons **Core** à l'intérieur duquel, vous allez ajouter une nouvelle classe **GameObject** :



Je vais rester très simple dans le code (ou essayer). Commençons par importer la base du Framework dans cette nouvelle classe :

```
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Graphics;
```

Vous pourrez remarquer qu'effectivement, MonoGame est une implémentation de XNA, on ne peut même pas le cacher, car on utilise l'espace de noms de XNA, ce qui permet d'avoir les codes XNA compatibles ! Puis dans la classe elle-même cette fois-ci, qu'est-ce qu'il nous faut ? Comme dit plus haut, tous les objets ont la même base à savoir… la position et l'image à afficher :

```
public Vector2 Position;
public Texture2D Texture;
```

Pour finir, il nous faut une méthode de dessin :

```
public void Draw(SpriteBatch spriteBatch)
{
    spriteBatch.Draw(Texture, Position,
Color.White);
}
```

Méthode dans laquelle nous appelons la méthode **Draw** de notre argument. Le dernier argument qu'on passe est une couleur qui permet de modifier la teinture de l'image. Nous utilisons ici la couleur blanche pour ne pas modifier notre image. Vous vous demandiez sûrement à quoi peut donc servir cet objet **SpriteBatch** ! Maintenant vous le savez, il permet de dessiner à l'écran des textures, 2D dans notre cas.

4.3. Les autres objets

Maintenant que nous disposons de notre classe de base, nous pouvons ajouter les différentes classes enfants dont nous avons besoin, à savoir : le monde, le héros Pacman et ses ennemis. Toujours dans le dossier **Core**, ajoutez les nouvelles classes nommées **World**, **Player**, **Enemies** (un fichier par classe) et faites-les hériter de notre classe **GameObject**.

5. Dessine-moi… un monde

Nous y sommes, nous allons ajouter notre image de fond

où notre héros évoluera ! Dans notre classe **MyPacman**, dans la partie des importations, ajoutez notre espace de noms **Pacman.Core** :

```
using Pacman.Core;
```

Juste après les deux constantes que nous avons ajoutées au début, il nous faut déclarer une variable pour notre monde donc :

```
World world;
```

Puis dans la méthode **Initialize** juste après le commentaire :

```
world = new World();
```

Ce qui nous crée une instance de notre image de fond. Maintenant, il faut ajouter l'image au projet, téléchargez l'archive où se trouvent les quelques images du projet : Lien 08
Décompressez-la puis faites un glisser-déposer de l'image **world.png** vers le dossier **Content** de notre projet dans l'explorateur de solutions. Le dossier **Content** permet d'avoir un accès direct à nos ressources dans le projet. Les ressources ainsi déposées sont automatiquement copiées dans le dossier physique du même nom dans le dossier de notre projet. En de plus de cela, le dossier est couplé au ContentManager qui va grandement nous faciliter les choses, du chargement des images à leur libération qui elle, est automatique !

L'exclusion d'une ressource du projet (menu contextuel avec le clic droit) ne la supprimera pas physiquement, elle est juste déréférencée de notre projet. Pour la supprimer définitivement, utilisez la commande **Supprimer** plus bas dans le menu contextuel.

Une chose importante à faire à l'ajout de chaque ressource : cliquez sur l'image et dans la fenêtre de propriétés, sur la propriété **Copier dans le répertoire de sortie**, mettez à la valeur **Copier si plus récent**. Sans cela, l'image ne sera pas copiée dans le répertoire de l'exécutable et il y aura une erreur de chargement !

Nous y sommes presque, il ne reste plus que deux étapes ! L'avant-dernière consiste à charger notre image fraîchement ajoutée à notre projet. Dans la méthode **LoadContent**, juste après le commentaire qui nous est destiné, ajoutez ces deux lignes :

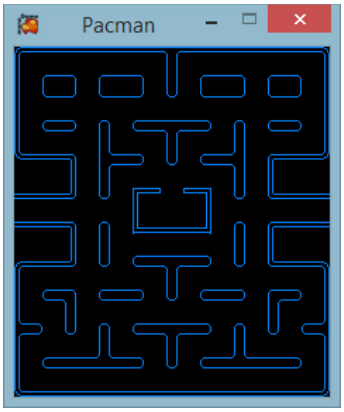
```
world.Texture = Content.Load<Texture2D>("world");
world.Position = new Vector2(0, 0);
```

Vous aurez pu remarquer que nous appelons notre image uniquement par son nom ! Oui étant donné que c'est le ContentManager qui se charge de la gestion des ressources, il ne nous est pas nécessaire d'utiliser l'extension. Cela ajoute cependant une contrainte, chaque fichier doit avoir un nom unique !

La seconde ligne nous permet de positionner l'image. Comme dans toutes les bibliothèques graphiques, ces coordonnées correspondent au coin supérieur gauche. Il ne nous reste plus qu'à dessiner notre image. Dans la méthode **Draw**, ajoutez ces lignes :

```
spriteBatch.Begin();
world.Draw(spriteBatch);
spriteBatch.End();
```

Je pense que vous vous demandez à quoi servent les appels qui entourent notre appel à la méthode **Draw** de notre objet ! Je ne vais pas trop rentrer dans les détails. Ce qu'il faut savoir, c'est que c'est ici que MonoGame va lancer un traitement qui sera spécifique à la signature de la méthode **Begin** utilisée, car il existe différentes signatures. Vous pouvez vous rendre sur le site de Microsoft pour plus d'informations à ce sujet SpriteBatch.Begin Méthode (Lien 09)
Dans cet appel, ce qu'il faut savoir, c'est que le rendu de nos images ne sera effectif qu'à l'appel de la méthode **End**. Une fois appelée, cette méthode va paramétrer la carte graphique et appellera chaque méthode **Draw** dans leur ordre d'apparition dans le code.
Lancez votre projet, vous devriez vous retrouver avec un écran similaire au mien :



6. Et Pacman alors ?

On y vient, un monde vide ne sert à rien, je le sais ! Nous allons donc maintenant nous préoccuper un peu de notre héros. Dans le dossier **Content**, ajoutez le jeu de sprites le représentant !
À partir d'ici, cela va se corser un petit peu. Lorsque nous regardons notre image, on voit qu'il y a toute une série d'images à la suite ! Ce que nous allons devoir faire, c'est dessiner la bonne portion par rapport à la direction dans laquelle le personnage se déplace et également afficher les diverses images intermédiaires qui permettront de créer une petite animation. Modifions la classe GameObject.
Commençons par ajouter quelques variables :

```
// Rectangle permettant de définir la zone de
l'image à afficher
public Rectangle Source;
// Durée depuis laquelle l'image est à l'écran
public float time;
// Durée de visibilité d'une image
public float frameTime = 0.1f;
// Indice de l'image en cours
public int frameIndex;
```

Ensuite, pour nous simplifier la tâche, nous allons définir en avance l'indice de chaque image. Nous allons faire ceci avec une énumération tout simplement. Il faut juste

prendre le temps de regarder l'image de notre héros :

Nous avons la direction droite à l'indice **0** et **1**, etc. Deux images par direction, ce qui nous donne le code suivant (il en sera de même pour les ennemis, ce qui nous arrange bien) :

```
public enum framesIndex
{
    RIGHT_1 = 0,
    RIGHT_2 = 1,
    BOTTOM_1 = 2,
    BOTTOM_2 = 3,
    LEFT_1 = 4,
    LEFT_2 = 5,
    TOP_1 = 6,
    TOP_2 = 7
}
```

Notez bien le fait que ce n'est pas le même nom que la variable, frame est au pluriel !

Ajoutons ensuite quelques propriétés qui ne possèdent que la méthode **Get**. Qui dit propriété dit variable privée en plus :

```
private int _totalFrames;
public int totalFrames
{
    get { return _totalFrames; }
}
private int _frameWidth;
public int frameWidth
{
    get { return _frameWidth; }
}
private int _frameHeight;
public int frameHeight
{
    get { return _frameHeight; }
}
```

Inutile de les commenter, je suppose, leur nom est assez explicite !

Donnez toujours des noms les plus explicites possible à vos variables, méthodes, propriétés, etc. Le code n'en sera que largement plus lisible et vous éviterez un tas de commentaires inutiles !

Nous allons maintenant ajouter deux constructeurs à notre classe. Jusque-là nous utilisions le constructeur par défaut, mais si on veut pouvoir initialiser nos propriétés par exemple, il nous faut construire nos propres constructeurs :

```
public GameObject()
{
}
public GameObject(int totalAnimationFrames, int
frameWidth, int frameHeight)
{
    _totalFrames = totalAnimationFrames;
    _frameWidth = frameWidth;
    _frameHeight = frameHeight;
}
```

Pour le moment on reste simple, le constructeur par défaut est juste là pour que Visual Studio ne nous embête pas, car il voudra un constructeur de ce type. Nous appellerons le second constructeur pour pouvoir commencer à régler nos animations. Ajoutons également une méthode **DrawAnimation** en dessous de **Draw** :

```
public void DrawAnimation(SpriteBatch
spriteBatch)
{
}
```

Nous reviendrons un peu plus tard sur notre méthode **DrawAnimation**. Il nous faut avant de l'utiliser, obtenir des informations supplémentaires comme, un **Rectangle** contenant les coordonnées du sprite à afficher. Il faut aussi calculer le temps passé afin de savoir quand changer l'indice de notre sprite permettant de calculer ses coordonnées. Ajoutons alors une méthode **UpdateFrame** (en dessous de **DrawAnimation**) :

```
public void UpdateFrame(GameTime gameTime)
{
    time +=
(float)gameTime.ElapsedGameTime.TotalSeconds;

    while (time > frameTime)
    {
        frameIndex++;
        time = 0f;
    }
    if (frameIndex > _totalFrames)
        frameIndex = 0;

    Source = new Rectangle(
        frameIndex * frameWidth,
        0,
        frameWidth,
        frameHeight);
}
```

Je sais que ça a l'air compliqué, mais il n'en est rien. Cette fonction permet dans un premier temps de calculer le temps passé depuis notre dernière mise à jour de l'affichage de notre sprite. Nous passons ensuite au prochain indice de notre image si nous avons dépassé le temps d'affichage puis, on remet à zéro la variable **time** pour la réutiliser correctement pour la prochaine mise à jour. Si l'indice dépasse le nombre de sprites dans notre collection, on repasse au premier. Nous calculons ensuite la position du nouveau sprite à afficher en déterminant sa position par rapport à l'indice en cours. Vous voyez, c'est simple comme bonjour !
Revenons maintenant à notre méthode **DrawAnimation**, nous avons maintenant toutes les informations nécessaires au bon affichage de notre héros. Ajoutons alors la méthode de rendu :

```
spriteBatch.Draw(Texture, Position, Source,
Color.White);
```

Avant de pouvoir lancer et même afficher tout cela, il ne faut pas oublier un détail, si vous regardez nos constructeurs de la classe **GameObject**, en pensant que

notre héros est animé, il faut également un constructeur pour notre personnage ! Dans notre classe **Player**, ajoutons un constructeur avec les mêmes paramètres que le second constructeur de notre classe **GameObject**. Ce constructeur appellera justement le constructeur parent grâce à l'instruction **base** :

```
public Player(int totalAnimationFrames, int
frameWidth, int frameHeight)
    : base(totalAnimationFrames, frameWidth,
frameHeight)
{
}
```

Retrouvons-nous maintenant dans la classe **MyPacman** pour y ajouter la déclaration, l'instanciation et les appels qui vont bien. En dessous de la déclaration de notre monde, ajoutons celle de notre personnage :

```
Player player;
```

Puis, dans la méthode **Initialize**, créons enfin notre héros ! Dans l'ordre des arguments, il possède huit images et fait treize pixels de largeur et de hauteur.

```
player = new Player(8, 13, 13);
```

Viens ensuite le chargement et le positionnement dans la méthode **LoadContent** :

```
player.Texture =
Content.Load<Texture2D>("pacman");
player.Position = new Vector2(0, 109);
```

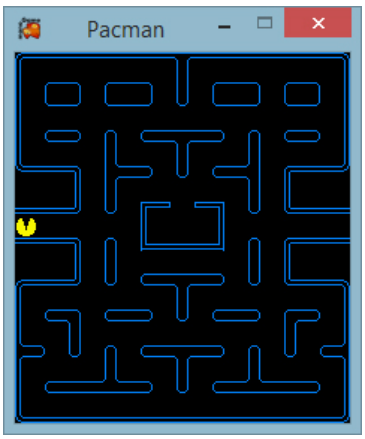
Dans la méthode **Update**, après la définition de la taille de la surface de jeu :

```
player.UpdateFrame(gameTime);
```

Et pour terminer le rendu dans la méthode **Draw** (bien sûr après l'affichage du monde, sinon il serait en dessous le pauvre):

```
player.DrawAnimation(spriteBatch);
```

Vous pouvez d'ores et déjà lancer le programme :



Bon vous me direz… Il a la tête qui tourne le pov' Pacman, hélas oui je vous répondrai ! Il ne sait pas encore où il faut aller ni même comment démarrer correctement et c'est ce que nous allons voir dans le prochain chapitre !

7. On va bouger, bouger…

Bon, notre personnage tourne sur lui-même, c'est pas sympa ça, nous allons donc y remédier. On va devoir travailler sur la modification de quelques classes donc chaque chapitre correspondra à la modification de l'une d'entre elles. Nous commencerons par la classe **GameObject** puis nous passerons à la classe **Player**. Nous retournerons sur la classe **GameObject** pour finir ses modifications et nous terminerons par la classe **MyPacman**. Je ne vous avais pas dit que j'écrivais ce tutoriel en même temps que je développe l'application ? Maintenant vous le savez ;)

7.1. Modification de la classe : GameObject

Qui dit mouvement dit direction, il nous faut donc des constantes pour nous simplifier la tâche plus tard, ajoutons une énumération :

```
public enum Direction
{
    LEFT = 0,
    RIGHT = 1,
    TOP = 2,
    BOTTOM = 3
}
```

On peut, par la même occasion, ajouter une variable de ce type :

```
public Direction direction;
```

7.2. Modification de la classe : Player

Ajoutons de quoi reconnaître le clavier :

```
using Microsoft.Xna.Framework.Input;
```

Puis ajoutons-y une méthode permettant de gérer les touches du clavier :

```
public void Move(KeyboardState state)
{
}
```

On va également initialiser la position de départ de notre personnage dans son constructeur ainsi que la première image à afficher (si Visual Studio vous embête à cause du type pour notre indice d'image ce n'est pas grave, nous procéderons à d'autres modifications plus loin) :

```
direction = Direction.RIGHT;
frameIndex = framesIndex.RIGHT_1;
```

Que devons-nous savoir quant aux mouvements de notre personnage ? Si on appuie sur la touche **Z**, alors il monte donc, **Q** pour la gauche, **S** pour le bas et **D** pour la droite ! N'oublions pas non plus de renseigner notre variable **direction** ! On va pour le moment bouger de un **pixel**. Nous pouvons donc implémenter notre méthode **Move** comme suit :

```
public void Move(KeyboardState state)
{
    if (state.IsKeyDown(Keys.Z))
    {
        direction = Direction.TOP;
        Position.Y -= 1;
    }
    if (state.IsKeyDown(Keys.Q))
    {
        direction = Direction.LEFT;
        Position.X -= 1;
    }
    if (state.IsKeyDown(Keys.S))
    {
        direction = Direction.BOTTOM;
        Position.Y += 1;
    }
    if (state.IsKeyDown(Keys.D))
    {
        direction = Direction.RIGHT;
        Position.X += 1;
    }
}
```

Vous savez maintenant comment gérer le clavier !

7.3. Modification de la classe : GameObject, la suite

Il nous faut finir les modifications de cette classe. Nous allons enfin utiliser nos constantes **framesIndex** mais avant cela, il va falloir changer le type de notre variable **frameIndex** qui est de type **int**,en type **framesIndex**.
Ensuite, supprimez l'incrémentation de notre **frameIndex**, oui oui, on le vire, y'en a marre de tourner en rond ! Ce que nous devons faire, c'est utiliser les bonnes images suivant l'orientation de notre personnage, orientation que nous définissons déjà au chargement de notre classe **Player**. Nous allons donc mettre un **switch** en place où chaque **case** correspond à ? Une direction !

```
switch (direction)
{
    case Direction.TOP:
        if (frameIndex == framesIndex.TOP_1)
            frameIndex = framesIndex.TOP_2;
        else
            frameIndex = framesIndex.TOP_1;
        break;
    case Direction.LEFT:
        if (frameIndex == framesIndex.LEFT_1)
            frameIndex = framesIndex.LEFT_2;
        else
            frameIndex = framesIndex.LEFT_1;
        break;
    case Direction.BOTTOM:
        if (frameIndex == framesIndex.BOTTOM_1)
            frameIndex = framesIndex.BOTTOM_2;
        else
            frameIndex = framesIndex.BOTTOM_1;
        break;
    case Direction.RIGHT:
        if (frameIndex == framesIndex.RIGHT_1)
            frameIndex = framesIndex.RIGHT_2;
        else
            frameIndex = framesIndex.RIGHT_1;
        break;
}
```

Ce que nous faisons ici, c'est déterminer à quel indice d'image correspond le **frameIndex** et nous ajustons la valeur en fonction du résultat ! Ensuite il faut modifier l'appel suivant de cette façon :

```
Source = new Rectangle (
    (int)frameIndex * frameWidth,
    0,
    frameWidth,
    frameHeight);
```

Nous avons juste ajouté un cast pour la position X sur notre série d'images. Normal, nous avons changé le type de notre variable ! Passons à la dernière classe à modifier !

7.4. Modification de la classe : MyPacman

Du vite fait dans cette classe. Il y a juste à ajouter dans la méthode **Update,** avant notre appel à **player.UpdateFrame**, notre méthode **Move**. Sans oublier de récupérer l'état du clavier et de le passer en paramètre :

```
player.Move(Keyboard.GetState());
```

Vous pouvez lancer le programme ! Miracle ! Ça bouge ! Notre personnage a pris vie, enfin ! Bien sûr, il bouge partout, même là où il ne peut normalement pas aller, mais ça, ce sera pour une prochaine fois !

8. Résumé

Oui je sais, c'est la fin, mais que de cette première partie !
Nous avons donc vu dans cette première partie, comment installer les outils nécessaires au développement de jeux avec MonoGame en C# sur Windows. Nous avons également vu comment, créer des classes adaptées pour ce type de projet, les informations dont nous avons besoin pour avoir un projet un minimum opérationnel, comment afficher une image fixe et comment également, créer et afficher une animation et pour finir, nous avons vu comment gérer les déplacements !
Cela fait déjà pas mal de choses pour un bon début !

9. Code source

Vous pouvez télécharger ci-après, le code source de la solution complète par rapport à cette première partie du tutoriel : Lien 10

Retrouvez l'article de Franck Hecht en ligne : Lien 11

Services à superviser avec Nagios - apprendre, aller plus loin avec les services

Afin de garantir le bon fonctionnement d'un parc informatique, il est nécessaire de le superviser et pour cela, nous choisissons Nagios. Mais une fois ce dernier installé, la question qui revient le plus souvent est : « que dois-je vraiment superviser ? ».
Dans ce tutoriel, nous listerons quelques services utiles à superviser et surtout, nous verrons comment le faire via la recherche ou le développement de plugins adéquats.

1. Introduction

Ce tutoriel n'a pas pour vocation de vous apprendre à installer Nagios. Pour cela, veuillez lire mon précédent article sur le sujet : Installation et configuration de Nagios pour débutants - Apprendre par l'exemple (Lien 12).
Superviser un parc informatique dans tous ses recoins est infini et je n'ai pas la prétention de vous montrer comment le faire. C'est juste un partage d'expérience qui vous permettra de gagner du temps par rapport aux différentes questions que l'on peut se poser et de pouvoir y répondre facilement.

2. Qu'est-ce qu'un service ?

Dans cet article, je ferai un abus de langage en utilisant le mot « service ». Surveiller un service revient à surveiller qu'une machine est joignable, vérifier depuis combien de temps elle est en fonctionnement, surveiller que notre DHCP ou DNS fonctionne, vérifier l'espace disque de nos serveurs, vérifier qu'ils sont à l'heure, vérifier la mémoire RAM…
Ce tutoriel contient une liste de services que nous surveillons (sur des machines Windows, Linux ou des Switches). Vous y trouverez le plugin utilisé pour ce faire et quelques explications.
Comme je vous l'ai dit ci-dessus, cette liste est non exhaustive et a pour but de vous montrer les services que j'avais besoin de surveiller en priorité.

3. Prérequis

Afin de partir sur de bonnes bases, je vous recommande sur tous vos serveurs Windows, Linux et Mac d'installer les outils ci-dessous.

3.1. Perl

Les plugins par défaut de Nagios ne permettent pas de superviser tout ce que l'on veut. Nous serons amenés de ce fait à soit télécharger de nouveaux plugins, soit à en développer nous-mêmes.
Beaucoup de ces programmes sont développés en Perl. Il est donc nécessaire d'installer Perl et certains de ses modules dont nous pourrions avoir besoin, notamment les modules **Nagios::Plugin** et **Net::SNMP**.
Sous Linux/Mac, Perl est déjà installé. Sous Windows, ce n'est pas le cas. Installez-le si besoin : Lien 13.
Sous Debian, il est possible de passer par les paquets de la distribution pour installer les modules :

Installation modules Perl
```
apt-get install libnagios-plugin-perl libnet-snmp-perl
```

Pour les autres distributions, un autre moyen simple, si vous n'avez pas de paquets, c'est d'utiliser l'utilitaire cpan déjà présent en « root » ou « sudo » :

cpan
```
cpan -i Nagios::Plugin Net::SNMP
```

Si vous rencontrez des soucis, voici une documentation pouvant vous aider : Installation des modules Perl CPAN (Lien 14). Et si cela ne suffit point, le forum Perl est à votre disposition : Lien 15.

3.2. SNMP

Pour obtenir certaines informations sur vos serveurs ou équipements réseau comme les switches, parfois, le seul moyen est d'utiliser SNMP. De ce fait, autant de suite l'installer ou l'activer sur le matériel à superviser.
En ce qui concerne les serveurs comme Debian, voici les paquets à installer :

Installation snmp
```
apt-get install snmp snmpd snmp-mibs-downloader
```

Vous aurez ensuite besoin de configurer SNMP afin de permettre au serveur de supervision d'interroger votre serveur. Ouvrez le fichier « /etc/snmp/snmpd.conf ».
Voici quelques modifications à faire.

Le fichier de configuration a évolué avec les versions récentes de Debian.

Voici les changements avant et après la version 5 de Debian :

Debian 5 et moins
```
# sec.name  source community
# com2sec paranoid default public
com2sec readonly  default public
#com2sec readwrite default private
```

Debian 6, 7 et plus
```
agentAddress udp:161
rocommunity public <IP ou NOM DNS serveur de supervision>
```

Vous pouvez bien sûr n'autoriser qu'une surveillance en local en mettant 127.0.0.1 ou localhost.

Redémarrer SNMP
```
/etc/init.d/snmpd restart
```

Petit test local :

test : snmpwalk
```
snmpwalk -c public -v 1 localhost | more
```

Pour approfondir vos connaissances sur SNMP, je vous conseille les deux articles suivants :

- Introduction à Net-SNMP et SNMP version 1 : Lien 16 ;
- Introduction à Net-SNMP - notifications et Trap en version 1 : Lien 17.

4. Rappels Nagios

Nagios est un moniteur de supervision qui nous alerte de toutes pannes ou anomalies rencontrées sur les serveurs ou tous équipements réseau que nous surveillons. Cette surveillance se fait à l'aide d'agents installés sur ces équipements. Dans ce tutoriel, nous allons surveiller certains services et utiliser certains agents.

4.1. Sous Windows

L'agent utilisé est l'un des plus répandus : « NSClient++ ». Il est dédié à l'environnement Windows et joue trois rôles essentiels :

- agent de supervision ;
- fonction de transport NRPE ;
- fonction de transport NSCA.

Dans ce tutoriel, nous utiliserons deux modes de fonctionnement de NSClient sur les trois existant.

4.1.1. Mode nsclient

Historiquement, c'est le premier mode de NSClient++ à sa création. En fait, il existait un agent nommé « nsclient » qui était interrogé via le plugin standard « check_nt » de Nagios (Nagios s'appelait autrefois « **NetSaint** »). Pour configurer ce mode (comme les autres), tout se fait dans le fichier de configuration NSC.ini dans la section [NSClient] ou [Settings]. Pour en savoir plus, lisez la documentation d'installation de Nagios (Lien 18).
À travers ce mode et le plugin « check_nt », nous avons la possibilité de surveiller facilement plusieurs services sous Windows (Version NSClient++, CPU, Uptime, espace disque, mémoire, services Windows, nombre d'utilisateurs…). On l'utilise tout au long de cet article.

4.1.2. Mode NRPE

Le mode NRPE a un très grand avantage, car il permet à l'administrateur de développer des plugins en divers langages (Visual Basics, Perl…) et de les déposer sur les machines à superviser. Ainsi, ils seront lancés par l'agent à travers ce mode après réception de la demande du serveur Nagios. Le mode NRPE de Nagios permet aussi d'interroger les différents modules de NSClient++ et d'encrypter les échanges.

Pour utiliser ce mode depuis le serveur Nagios, on utilise le plugin « check_nrpe ». On l'utilise dans ce tutoriel, surtout pour surveiller des machines Linux.

4.1.3. Mode NSCA

Ce mode utilise le protocole NSCA que nous n'utiliserons pas dans ce tutoriel.

4.2. Sous Linux

Nous utiliserons la plupart du temps le protocole NRPE pour échanger avec les serveurs surveillés et y lancer des programmes locaux via le plugin « check_nrpe ». Parfois, nous utiliserons SNMP soit via le plugin Nagios « check_snmp » ou via des plugins personnels SNMP.

4.3. Équipements réseau (switches…)

Le seul moyen de superviser ces équipements est d'utiliser SNMP. Il faudra juste s'assurer qu'il est bien activé. Le plugin Nagios « check_snmp » ou des plugins personnels SNMP seront nécessaires.

4.4. Généralité

Dans tous les cas, il sera nécessaire sur le serveur de supervision de modifier ou non le fichier « /usr/local/nagios/etc/objects/commande.cfg » afin de créer une nouvelle commande. C'est souvent le cas pour de nouveaux plugins utilisant SNMP.
Lorsque la communication se fait via NRPE, il est nécessaire de faire une modification dans le fichier « /usr/local/nagios/etc/nrpe.cfg » sur le serveur Linux à surveiller pour configurer la ligne de commande.
Une fois ces améliorations apportées, il ne reste plus qu'à définir ses services pour chaque serveur et matériel à superviser. Vous verrez dans les exemples ci-dessous différentes écritures pour faire passer les arguments. Soyez attentif !
Je vous recommande un peu de lecture pour en savoir plus sur Nagios, NSClient++…

- NSClient++ - documentation française : Lien 19.
- NSClient++ - documentation officielle : Lien 20.
- Installation et configuration de Nagios pour débutants : Lien 21.

5. Services à surveiller

5.1. Version NSCLIENT++

Nous supervisons des machines Windows et nous avons besoin d'être sûrs qu'elles utilisent toutes la même version de l'agent NSCLIENT++.

C'est un choix personnel d'uniformiser les versions d'agents utilisés au sein d'un même parc informatique. Cela permet de minimiser des erreurs d'incompatibilités ou des soucis de maintenance.

C'est assez simple à mettre en place, car par défaut, c'est dans la configuration de base de Nagios. Tout se fait via « check_nt ».
Dans le fichier de configuration du serveur (fichier .cfg), voici le service à déclarer :

Version NSCLIENT++

```
# Version NSCLIENT
define service {
    use         generic-service
    host-name   SERVEURWINDOWS
    service_description    Version NSCLIENT++
    check_command    check_nt!CLIENTVERSION
}
```

Nagios affichera la version de votre client NSCLIENT++ du serveur Windows. Ce qui peut être intéressant, c'est de s'assurer que nos machines Windows ont la même version de l'agent. Pour cela, « check_nt » dispose d'une option « -l » pour sa variable CLIENTVERSION où l'on précise la version. Ainsi, si la requête n'a pas exactement la même réponse, vous aurez un warning dans Nagios.

Version NSCLIENT++

```
# Version NSCLIENT
define service {
    use         generic-service
    host-name   SERVEURWINDOWS
    service_description    Version NSCLIENT++
    check_command    check_nt!CLIENTVERSION!-l
"NSCLIENT++ 0.3.9.327 2011-08-16"
}
```

Sur la ligne check_command, on demande à Nagios d'utiliser le plugin « check_nt » avec deux arguments (« ! » est un séparateur d'arguments). Le premier argument est la variable et le deuxième la valeur à tester.
Maintenant, vous saurez si vos machines Windows ont cette version de NSCLIENT++ ou non.
Pour effectuer le test en ligne de commande,

Ligne de commande

```
root@supervision:~#
/usr/local/nagios/libexec/check_nt -H
SERVEURWINDOWS -v CLIENTVERSION -s MOTDEPASSE -p
12489 -l "NSCLIENT++ 0.1.1"
Mauvaise version du client utilisée: NSCLIENT++
0.3.9.327 2011-08-16, nécessaire:  NSCLIENT++
0.1.1
```

Vous voyez ci-dessus un exemple de résultat en cas de résultat négatif.

5.2. Espace disque

La surveillance de l'espace disque de vos serveurs est primordiale et Nagios a déjà tout prévu.

5.2.1. Windows

Nous utilisons « check_nt » avec la variable « USEDDISKSPACE ».
En ligne de commande, depuis votre serveur « supervision », voici la commande à lancer :

Windows : Espace disque - ligne de commande

```
root@supervision:~#
/usr/local/nagios/libexec/check_nt -H
SERVEURWINDOWS -v USEDDISKSPACE -s MOTDEPASSE -p
12489 -l C -w 80 -c 90 -u GB
C:\ - total: 30,01 Gb - utilisé: 19,75 Gb (66%) -
libre 10,25 Gb (34%) | 'C:\ Espace
Utilisé'=19,75Gb;24,00;27,01;0.00;30,01
```

Explication des options de la commande :

- -H : préciser le serveur Windows à superviser ;
- -v : variable pour « check_nt ». On utilise USEDDISKSPACE pour vérifier l'espace disque ;
- -s : mot de passe configuré dans NSCLIENT++ de la machine Windows ;
- -p : port par défaut ;
- -l : pour passer les arguments :
    - C : lecteur disque à surveiller ;
    - -w 80 : au-dessus de 80 % d'occupation, Nagios émet un Warning ;
    - -c 90 : au-dessus de 90 % d'occupation, Nagios émet une alerte critique ;
    - -u GB : permet juste de forcer un affichage avec la taille en GB.

Dans le fichier de configuration du serveur (fichier .cfg), voici le service à déclarer :

Service Windows

```
# Espace disque

define service{
    use             generic-service
    host_name         SERVEURWINDOWS
    service_description    C:\ Espace Disque
    check_command         check_nt!USEDDISKSPACE!-
l c -w 80 -c 90 -u GB
    }
```

Le mot de passe NSCLIENT++ n'est pas renseigné, car il l'est déjà dans le fichier « /usr/local/nagios/etc/objects/commands.cfg » comme expliqué dans la documentation d'installation de Nagios.

5.2.2. Linux

Nous utilisons « check_nrpe » et « check_disk ».
En ligne de commande, depuis votre serveur « supervision », voici la commande à lancer :

Linux : espace disque - ligne de commande

```
root@supervision:/usr/local/nagios/etc#/usr/local
/nagios/libexec/check_nrpe -H SERVEURLINUX -c
check_disk -a 20% 10% /
DISK OK - free space: / 38858 MB (84%
inode=94%);| /=6993MB;38644;43474;0;48305
```

La partition « / » est vérifiée et si le pourcentage d'espace disque restant est inférieur à 20 %, on a un warning et une alerte critique si inférieur à 10 %. En ligne de commande, l'option « -c » permet de spécifier le programme distant à lancer. L'option « -a » permet de passer les arguments.

Serveur Linux

```
# Vérification de l'espace disque
define service{
    use                          generic-service
    host_name                      SERVEURLINUX
    service_description            Espace disque /
    check_command                check_nrpe!
check_disk!20%!10%!/
    }
```

« ! » est un séparateur d'arguments.

5.3. CPU

Il est possible de vérifier la charge moyenne système durant les x dernières minutes. Tout est déjà présent dans Nagios par défaut.

5.3.1. Windows

Windows
```
# Charge CPU

define service{
    use             generic-service
    host_name         SERVEURWINDOWS
    service_description    CPU Load
    check_command        check_nt!CPULOAD!-l
5,80,90
    }
```

La vérification sera faite pour les cinq dernières minutes avec une limite entre 80 % et 90 % de charge CPU.

5.3.2. Linux

C'est un peu particulier. En fait, la charge CPU s'exprime normalement sans unité et correspond au nombre de processus en train d'utiliser le ou les processeurs de la machine.
Les utilitaires comme « uptime » ou « top » donnent cette information sous forme de trois valeurs inférieures à 1. Exemple : 0.15, 0.20 et 0.17.
Ces valeurs représentent la charge moyenne au cours de la dernière, des cinq et quinze dernières minutes.
Il faut considérer une machine chargée si la valeur est supérieure à « 1 » pour un serveur monoprocesseur ou « 2 » pour un biprocesseur…

Linux
```
# Vérification de la charge CPU

define service{
    use                     generic-service
    host_name               SERVEURLINUX
    service_description     Charge CPU
    check_command           check_nrpe!
check_load!0.7,0.6,0.5!0.9,0.8,0.7
    }
```

Voici une explication de notre service : un warning est émis par Nagios si la charge CPU excède 70 %, 60 %, 50 % de charge CPU les 1, 5 et 15 dernières minutes. Puis, une alerte critique au-delà de 90 %, 80 % et 70 % les 1, 5 et 15 dernières minutes.

5.4. Mémoire RAM

5.4.1. Windows

La vérification de la mémoire RAM est possible depuis Nagios et le plugin de vérification est déjà présent pour la surveillance des machines sous Windows. Voici un exemple de ligne de commande à lancer depuis votre serveur « supervision » :

Windows : MEMUSE
```
root@supervision:#
/usr/local/nagios/libexec/check_nt -H
SERVEURWINDOWS -v MEMUSE -s MOTDEPASSE -p 12489
-w 80 -c 90
Mémoire utilisée: total:2168,51 Mb - utilisée:
466,13 Mb (21%) - libre: 1702,39 Mb (79%) |
'Mémoire
utilisée'=466,13Mb;0,00;0,00;0.00;2168,51
```

Cette commande demande au serveur Windows à travers NSCLIENT++ de vérifier la consommation de la RAM (variable MEMUSE de check_nt), de retourner une alerte (warning) si l'utilisation dépasse 80 % ou une alerte critique si elle dépasse 90 %. Pour Nagios, voici la configuration du service :

Windows : service MEMUSE
```
# Mémoire RAM

define service{
    use             generic-service
    host_name         SERVEURWINDOWS
    service_description    Mémoire utilisée
    check_command        check_nt!MEMUSE!-w 80 -c
90
    }
```

5.4.2. Linux

Il n'y a pas de plugin installé par défaut sur Nagios pour surveiller la mémoire RAM. Pour ce faire, il va falloir soit développer son propre plugin, soit en trouver un sur la toile comme expliqué ici : Lien 22.
En ce qui me concerne, pour la gestion de la mémoire, j'ai créé mon propre plugin que j'ai nommé « check_memory.pl ». Ce dernier fonctionne sous Linux, Mac et sous Windows, mais je ne l'utilise que sous Linux :

check_memory.pl
```
#!/usr/bin/perl
#================================================
# Auteur : djibril
# Date   : 29/10/2013
# But    : Check Memory and Swap on Linux/Unix
#================================================
use strict;
use warnings;

use Nagios::Plugin;
use English '-no_match_vars';
use Sys::MemInfo qw/availkeys/;
use vars qw/ $VERSION /;

# Version du plugin
$VERSION = '1.0';

my $LICENCE = 'This Plugin is free';

my $plugin_nagios = Nagios::Plugin->new(
    shortname => 'Check memory',
    usage     => 'Usage : %s [ -c|--
critical=<threshold> ] [ -w|--warning=<threshold>
] [ -M|--memory=<mem or swap> ]',
    version   => $VERSION,
    license   => $LICENCE,
);

if ( lc $OSNAME eq 'mswin32' ) {
    $plugin_nagios->nagios_exit( CRITICAL,
"[Win32] This plugin does not work for this
plateform\n" );
}
```

```
# Activer gestion mémoire swap
$plugin_nagios->add_arg(
    spec     => 'memory|M=s',
    help     => 'type of memory : swap or mem',
    required => 1,
);

# Définition de l'argument warning
$plugin_nagios->add_arg(
    spec     => 'warning|w=f',
# Nous acceptons des nombres réels
    help     => 'Exit with WARNING status if more
than pourcentage',
    required => 1,
);

# Définition de l'argument critical
$plugin_nagios->add_arg(
    spec     => 'critical|c=f',
    help     => 'Exit with CRITICAL status if
more than pourcentage',
    required => 1,
);

# Activer le parsing des options de ligne de
commande
$plugin_nagios->getopts;
my %memory = map { $_ => Sys::MemInfo::get($_) }
availkeys();

my $percent_memory_used = $plugin_nagios->opts-
>memory eq 'swap' ? sprintf '%.2f',
(($memory{totalswap} - $memory{freeswap})/
$memory{totalswap}) * 100
    : sprintf '%.2f', (($memory{totalmem} -
$memory{freemem})/$memory{totalmem}) * 100
    ;

my $message_memory = $plugin_nagios->opts->memory
eq 'swap' ?
      'Total Memory Swap : ' . sprintf('%.2f',
($memory{totalswap}/(1024*1024))) . 'MB - Swap
used : ' . $percent_memory_used . '%'
    : 'Total Memory : ' . sprintf('%.2f',
($memory{totalmem}/(1024*1024))) . 'MB - Mem used
: ' . $percent_memory_used . '%'
    ;
if ( $plugin_nagios->opts->verbose ) {
    while ( my ($key, $value) = each %memory ) {
        print "$key : $value\n";
    }
}

my $code_retour = $plugin_nagios-
>check_threshold(
    check    => $percent_memory_used,
    warning  => $plugin_nagios->opts->warning,
    critical => $plugin_nagios->opts->critical,
);

$plugin_nagios->nagios_exit( $code_retour,
$message_memory );
```

Ce programme permet de vérifier la mémoire RAM ou la mémoire Swap. Il faut le déposer dans le répertoire « /usr/local/nagios/libexec/ » du serveur Linux à superviser. Rendez-le exécutable.

chmod

```
chmod +x check_memory.pl
```

Ce programme n'est fonctionnel que si les modules Perl suivants sont installés sur le serveur :

Installation Modules Perl

```
cpan -i Nagios::Plugin Sys::MemInfo
```

Vous pouvez tester le programme sur le serveur en ligne de commande :

```
# /usr/local/nagios/libexec/check_memory.pl -v -w
85 -c 95 -M mem
totalmem : 2124271616
freeswap : 2221649920
totalswap : 2222977024
freemem : 354693120
Check memory OK - Total Memory : 2025.86MB - Mem
used : 83.30%

# /usr/local/nagios/libexec/check_memory.pl -w 10
-c 20 -M swap
Check memory OK - Total Memory Swap : 2120.00MB -
Swap used : 0.06%
```

Il faut maintenant configurer NRPE du serveur à superviser pour que ce programme puisse être interrogé à distance.

NRPE

```
# vi /usr/local/nagios/etc/nrpe.cfg
…
…
command[check_memory]=/usr/local/nagios/libexec/c
heck_memory.pl -w $ARG1$ -c $ARG2$ -M $ARG3$
```

Les trois arguments sont obligatoires pour notre programme.
Faisons un test en ligne de commande pour utiliser NRPE en local sur le serveur à superviser et depuis le serveur Nagios.

Serveur Linux

```
# /usr/local/nagios/libexec/check_nrpe -H
127.0.0.1 -c check_memory -a 80 90 swap
Check memory OK - Total Memory Swap : 2120.00MB -
Swap used : 0.06%
```

Serveur supervision Nagios

```
root@supervision:~#
/usr/local/nagios/libexec/check_nrpe -H
SERVEURLINUX -c check_memory -a 80 90 mem
Check memory WARNING - Total Memory : 2025.86MB -
Mem used : 82.73%
```

Tout fonctionne correctement !
Créons donc le service sur le serveur « supervision » pour que Nagios puisse surveiller la mémoire comme un grand !

Service

```
# Mémoire RAM

define service{
    use                     generic-service
    host_name               SERVEURLINUX
    service_description     Mémoire RAM
```

```
    check_command           check_nrpe!
check_memory!90!95!mem
    }

# SWAP

define service{
    use                     generic-service
    host_name               SERVEURLINUX
    service_description     Mémoire Swap
    check_command           check_nrpe!
check_memory!10!20!swap
    }
```

5.5. Uptime

Le terme « uptime » correspond au temps depuis lequel une machine est en fonctionnement. Le redémarrage de celle-ci réinitialise ce temps à zéro. De mon avis personnel, il peut être intéressant de savoir depuis combien de temps un serveur n'a pas été redémarré et de générer une alerte au bout d'un certain temps (un mois ou deux). Redémarrer un serveur peut permettre de vider la mémoire RAM, les caches, de faire une vérification de disques…

5.5.1. Windows

Via « check_nt » et « NSCLIENT++ », il est possible d'avoir l'uptime. Malheureusement, nous n'avons pas la possibilité d'obtenir une alerte à partir d'un seuil défini, car NSClient++ ne donne uniquement que cette durée et ne fait aucune comparaison, donc n'envoie aucune alerte de type warning ou critique à Nagios. Pourtant, nous souhaitons avoir une alerte en fonction du temps écoulé, ainsi, check_nt ne nous aide pas.
Il existe un plugin « CheckUpTime » déjà disponible dans NSCLIENT++ qu'il faudra interroger par « NRPE ». Mais, pour utiliser NRPE à travers NSCLIENT++, il faut l'activer. Sous le PC Windows, dans le fichier de configuration NSC.ini, dans la section [NRPE], il faudra décocher la ligne :

NSC.ini
```
;allow_arguments=0
```

Nous devons avoir :

activer NRPE
```
allow_arguments=1
```

Penser à redémarrer le service NSCLIENT++.
Maintenant, nous pouvons depuis notre serveur « supervision » lancer une requête.

Check_nt : pas de seuil
```
root@supervision:~#
/usr/local/nagios/libexec/check_nt -H
SERVEURWINDOWS -v UPTIME -p 12489 -s MOTDEPASSE
Système démarré - 9 jour(s) 5 heure(s) 10
minute(s)
```

Maintenant, utilisons « CheckUpTime » pour émettre un warning au-dessus de huit jours et critical au-dessus de dix jours :

CheckUpTime
```
root@supervision:~#
/usr/local/nagios/libexec/check_nrpe -H
SERVEURWINDOWS -c CheckUpTime -a MaxCrit=10d
MaxWarn=8d
WARNING: uptime: 1w 2d 5:28 >
warning|'uptime'=797289000;777600000;864000000
```

Le choix du seuil de huit à dix jours est personnel et à titre d'exemple pour ce tutoriel. Ce n'est en aucun cas une règle !

Tout fonctionne bien, configurons le service sous Nagios :

Service
```
# Temps de fonctionnement du système

define service{
 use             generic-service
 host_name         SERVEURWINDOWS
 service_description    Vérification Uptime
 check_command        check_nrpe!CheckUpTime!
MaxCrit=60d MaxWarn=30d
 }
```

Pour en savoir plus sur toutes les options de « CheckUpTime », n'hésitez pas à consulter le site officiel de NSCLient++ : Lien 23.

5.5.2. Linux

Il n'existe pas de plugin par défaut pour gérer l'uptime, il faut donc le concevoir soi-même ou en trouver un sur la toile.
Je vous propose un plugin que j'ai conçu dont le fonctionnement est le suivant : le programme installé sur le serveur client va récupérer l'uptime dans le fichier « /proc/uptime ». Ensuite, à partir du nombre de secondes, il va donner la possibilité de gérer les seuils d'alerte et de proposer un affichage personnalisé.

Nous aurions pu utiliser l'utilitaire « uptime » déjà présent sur le système puis parser son résultat, mais ce dernier formate déjà le temps, or nous avons besoin de gérer les seuils.

Voici le programme :

Plugin check_uptime pour Linux
```
#!/usr/bin/perl
#===============================================
# Auteur : djibril
# Date   : 31/10/2013
# But    : Check Uptime on Linux/Unix
#===============================================
use strict;
use warnings;

use Nagios::Plugin;
use English '-no_match_vars';
use vars qw/ $VERSION /;

# Version du plugin
$VERSION = '1.0';

my $LICENCE = 'This Plugin is free';

my $plugin_nagios = Nagios::Plugin->new(
    shortname => 'Check uptime',
```

```
    usage     => 'Usage : %s [ -c|--
critical=<threshold> ] [ -w|--warning=<threshold>
]',
    version   => $VERSION,
    license   => $LICENCE,
);

if ( lc $OSNAME eq 'mswin32' ) {
    $plugin_nagios->nagios_exit( CRITICAL,
"[Win32] This plugin does not work for this
plateform\n" );
}

# Définition de l'argument warning
$plugin_nagios->add_arg(
    spec     => 'warning|w=f',
    help     => 'Exit with WARNING status if more
than x days',
    label    => 'Days',
    required => 0,
);

# Définition de l'argument critical
$plugin_nagios->add_arg(
    spec     => 'critical|c=f',
    help     => 'Exit with CRITICAL status if
more than x days',
    label    => 'Days',
    required => 0,
);

# Activer le parsing des options de ligne de
commande
$plugin_nagios->getopts;

my $uptime_linux = uptime();

if ( !$uptime_linux ) {
    $plugin_nagios->nagios_exit( CRITICAL,
"unable to get uptime\n" );
}

my $seuil_warning  = $plugin_nagios->opts-
>warning ? $plugin_nagios->opts->warning * 60 *
60 * 24 : undef;
my $seuil_critical = $plugin_nagios->opts-
>critical ? $plugin_nagios->opts->critical * 60 *
60 * 24 : undef;
if ( $plugin_nagios->opts->verbose ) {
    print "Uptime : $uptime_linux\n";
    if ( defined $seuil_warning ) {
        print 'WARNING threshold : ',
$seuil_warning,
            ' (', time_format( $seuil_warning ),
')', "\n";
    }
    if ( defined $seuil_critical ) {
        print 'CRITICAL threshold : ',
$seuil_critical,
            '( ', time_format( $seuil_critical ),
')', "\n";
    }

}

my $code_retour = $plugin_nagios-
>check_threshold(
    check    => $uptime_linux,
    warning  => $seuil_warning,
    critical => $seuil_critical,
);
```

```
$plugin_nagios->nagios_exit( $code_retour,
'Uptime - ' . time_format($uptime_linux) );

sub uptime {
    open my $fh, '<', '/proc/uptime'
      or $plugin_nagios->nagios_exit( CRITICAL,
"/proc/uptime not exists, unable to get uptime\n"
), return;
    my ( $uptime, undef ) = split / /, <$fh>;
    close $fh;
    return $uptime;
}

sub time_format {
    my $totalsecondes = shift;

    if ( not defined $totalsecondes ) {
        $plugin_nagios->nagios_exit( CRITICAL,
"uptime not found" );
    }
    return '0 seconde' if ( $totalsecondes ==
0 );

    my $message = '';
    my ( $nbr_annees, $nbr_mois, $nbr_jours,
$nbr_heures, $nbr_minutes, $nbr_secondes ) = ();

    # Annees
    my $duree_an = 60 * 60 * 24 * 30.41 * 12;
    $nbr_annees = int( $totalsecondes / $duree_an
);
    if ( $nbr_annees > 0 ) {
        $totalsecondes = $totalsecondes -
( int( $totalsecondes / $duree_an ) *
$duree_an );
        $message .= ( $nbr_annees == 1 ) ?
"$nbr_annees year " : "$nbr_annees years ";
    }

    # Mois
    my $duree_mois = 60 * 60 * 24 * 30.41;
    $nbr_mois = int( $totalsecondes / $duree_mois
);
    if ( $nbr_mois > 0 ) {
        $totalsecondes = $totalsecondes -
( int( $totalsecondes / $duree_mois ) *
$duree_mois );
        $message .= ( $nbr_mois == 1 ) ?
"$nbr_mois month " : "$nbr_mois months ";
    }

    # Jours
    my $duree_jours = 60 * 60 * 24;
    $nbr_jours = int( $totalsecondes /
$duree_jours );
    if ( $nbr_jours > 0 ) {
        $totalsecondes = $totalsecondes -
( int( $totalsecondes / $duree_jours ) *
$duree_jours );
        $message .= ( $nbr_jours == 1 ) ?
"$nbr_jours day " : "$nbr_jours days ";
    }

    # Heures
    my $duree_heures = 60 * 60;
    $nbr_heures = int( $totalsecondes /
$duree_heures );
    if ( $nbr_heures > 0 ) {
        $totalsecondes = $totalsecondes -
( int( $totalsecondes / $duree_heures ) *
$duree_heures );
```

```
        $message .= ( $nbr_heures == 1 ) ?
"$nbr_heures hour " : "$nbr_heures hours ";
    }

    # Minutes
    my $duree_minutes = 60;
    $nbr_minutes = int( $totalsecondes /
$duree_minutes );
    if ( $nbr_minutes > 0 ) {
        $totalsecondes = $totalsecondes -
( int( $totalsecondes / $duree_minutes ) *
$duree_minutes );
        $message .= ( $nbr_minutes == 1 ) ?
"$nbr_minutes minute " : "$nbr_minutes minutes ";
    }

    # Secondes
    $totalsecondes = int $totalsecondes;
    if ( $totalsecondes > 0 ) {
        $message .= ( $totalsecondes == 1 ) ?
"$totalsecondes seconde " : "$totalsecondes
secondes ";
    }

    return $message;
}
```

Ce programme doit être déposé dans le répertoire « /usr/local/nagios/libexec ». Rendez-le exécutable.

chmod

```
chmod +x check_uptime.pl
```

Ce programme ne sera fonctionnel que si le module Perl suivant est installé sur le serveur :

Instalation module Perl

```
cpan -i Nagios::Plugin
```

Vous pouvez tester le programme sur le serveur en ligne de commande :

Commande

```
root@SERVEURLINUX:~#
/usr/local/nagios/libexec/check_uptime.pl -w 30
-c 60
Check uptime WARNING - Uptime - 1 month 11 days 5
hours 36 minutes 21 secondes
```

Il faut maintenant configurer NRPE sur le serveur à superviser pour que ce programme puisse être interrogé à distance.

NRPE serveur à superviser

```
# vi /usr/local/nagios/etc/nrpe.cfg
…
…
command[check_uptime]=/usr/local/nagios/libexec/c
heck_uptime.pl -w $ARG1$ -c $ARG2$
```

Les deux arguments sont obligatoires pour notre programme.
Faisons un test en ligne de commande pour utiliser NRPE en local sur le serveur à superviser et depuis le serveur Nagios.

NRPE check_uptime

```
# /usr/local/nagios/libexec/check_nrpe -H
127.0.0.1 -c check_uptime -a 30 60
Check uptime WARNING - Uptime - 1 month 11 days 5
hours 45 minutes 59 secondes
```

Supervision depuis serveur Nagios

```
root@supervision:~#
/usr/local/nagios/libexec/check_nrpe -H
SERVEURLINUX -c check_uptime -a 30 60
Check uptime WARNING - Uptime - 1 month 11 days 5
hours 47 minutes 34 secondes
```

Tout fonctionne correctement !
Créons le service sur le serveur « supervision » pour que Nagios puisse surveiller l'uptime !

Service check_uptime

```
# Vérification Uptime
define service{
    use                     generic-service
    host_name               SERVEURLINUX
    service_description     Vérification Uptime
    check_command           check_nrpe!
check_uptime!30!60
}
```

5.6. Vérifier des processus et services (Windows)

Sur certains serveurs, vous avez souvent des applications métier ou du moins des logiciels importants qui doivent absolument fonctionner. Le seul moyen est, bien souvent, de vérifier manuellement qu'un processus, qu'un service Windows est bien en fonctionnement (**ps -aux** sous Linux, **services** sous Windows). Avec Nagios, il est possible de s'en assurer de manière automatique.

5.6.1. Windows

Les variables PROCSTATE et SERVICESTATE nous permettent de vérifier qu'un processus et qu'un service Windows fonctionnent.

PROCSTATE

```
root@supervision:~#
/usr/local/nagios/libexec/check_nt -H
SERVEURWINDOWS -v PROCSTATE -s MOTDEPASSE -p
12489
Pas de service/processus spécifié
root@supervision:~#
/usr/local/nagios/libexec/check_nt -H
SERVEURWINDOWS -v PROCSTATE -s MOTDEPASSE -p
12489 -d SHOWALL -l toto
 toto: not running
root@supervision:~#
/usr/local/nagios/libexec/check_nt -H
SERVEURWINDOWS -v PROCSTATE -s MOTDEPASSE -p
12489 -d SHOWALL -l explorer.exe
 Explorer.EXE: Running
root@supervision:~#
/usr/local/nagios/libexec/check_nt -H
SERVEURWINDOWS -v PROCSTATE -s MOTDEPASSE -p
12489 -l explorer.exe
 OK: All processes are running.
root@supervision:~#
/usr/local/nagios/libexec/check_nt -H
SERVEURWINDOWS -v PROCSTATE -s MOTDEPASSE -p
12489 -l toto
 toto: not running
root@supervision:~#
/usr/local/nagios/libexec/check_nt -H
```

```
SERVEURWINDOWS -v PROCSTATE -s MOTDEPASSE -p
12489 -l explorer.exe,mysql.exe
 mysql.exe: not running
root@supervision:~#
/usr/local/nagios/libexec/check_nt -H
SERVEURWINDOWS -v PROCSTATE -s MOTDEPASSE -p
12489 -d SHOWALL -l explorer.exe,mysql.exe
 Explorer.EXE: Running - mysql.exe: not running
```

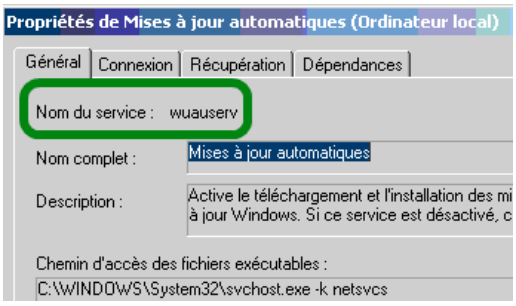
On comprend rapidement qu'il est simple de vérifier qu'un processus tourne ou non. L'option « -d SHOWALL » permet d'affiner l'affichage du résultat.

SERVICESTATE

```
root@supervision:~#
/usr/local/nagios/libexec/check_nt -H
SERVEURWINDOWS -v SERVICESTATE -s MOTDEPASSE -p
12489 -d SHOWALL -l "wuauserv"
wuauserv: Started
root@supervision:~#
/usr/local/nagios/libexec/check_nt -H
SERVEURWINDOWS -v SERVICESTATE -s MOTDEPASSE -p
12489 -d SHOWALL -l "OCS Inventory Service"
 OCS Inventory Service: Started
root@supervision:~#
/usr/local/nagios/libexec/check_nt -H
SERVEURWINDOWS -v SERVICESTATE -s MOTDEPASSE -p
12489 -d SHOWALL -l "toto"
 toto: Not found
```

De la même manière que PROCSTATE, la vérification est simple.

On renseigne le nom du service (nom logique) et non le « nom complet » du service (nom localisé (Ce nom localisé est issu de la traduction de Windows en fonction de la version, de la langue)). Voici une image montrant la propriété du service de mises à jour de Windows. On distingue bien le « Nom du service » et le « Nom complet » de ce dernier.



Windows : service MEMUSE

```
define service{
    use             generic-service
    host_name         SERVEURWINDOWS
    service_description     Explorer
    check_command         check_nt!PROCSTATE!-d
SHOWALL -l Explorer.exe
    }

define service{
    use             generic-service
    host_name         SERVEURWINDOWS
    service_description     MySQL
    check_command         check_nt!SERVICESTATE!-d
SHOWALL -l "MySQL"
}
```

5.6.2. Linux

Il n'y a pas de plugin installé par défaut sur Nagios. Créons un nouveau plugin que voici :

check_process.pl

```
#!/usr/bin/perl
#===============================================
# Auteur : Djibril
# But    : Plugin nagios permettant de vérifier
l'existence d'un
#          processus via le nom et/ou la commande
système
# Date   : 31/10/2013
#===============================================

use warnings;
use strict;
use Nagios::Plugin;
use Proc::ProcessTable;
use vars qw/ $VERSION /;

# Version du plugin
$VERSION = '1.0';

my $LICENCE = 'This Plugin is free';
my $plugin_nagios = Nagios::Plugin->new(
    shortname => 'Check process',
    usage     => 'Usage: %s [ -proc_name|-
n=<Process name> ] [ -path|-p=<Path> ]',
    version   => $VERSION,
    license   => $LICENCE,
);

# Définition des options de ligne de commande
# 1- Récupération du nom du processus
$plugin_nagios->add_arg(
    spec     => 'proc_name|n=s',
    help     => 'Process name',
    label    => 'STRING',
    required => 1,
);

# 2- CmdLine du processus
$plugin_nagios->add_arg(
    spec     => 'path|p=s',
    help     => 'Command line of process',
    label    => 'STRING',
    required => 0,
);

# Parser les options
$plugin_nagios->getopts;

# Recherche du processus
my $process_table = new Proc::ProcessTable;

my $proc_name = $plugin_nagios->opts->proc_name;

PROC:
foreach my $proc ( @{ $process_table->table } ) {
    if ( $proc->fname eq $proc_name ) {
        my $message_OK = "Process $proc_name (" .
$proc->pid . ') state (' . $proc->state . ')';

        # Path definie par nagios
        if ( my $path = $plugin_nagios->opts-
>path and $plugin_nagios->opts->path !~
m{^\s*$} ) {
```

```
            if ( $proc->cmndline ne $path ) {
                $plugin_nagios-
>nagios_exit( WARNING, "Process $proc_name exists
but not path : $path" );
                last PROC;
            }
            $plugin_nagios->nagios_exit( OK,
$message_OK );
        }
        $plugin_nagios->nagios_exit( OK,
$message_OK );
    }
}

$plugin_nagios->nagios_exit( CRITICAL,
"$proc_name not found" );
```

Ce programme doit être déposé dans le répertoire « /usr/local/nagios/libexec ». Rendez-le exécutable.

chmod

```
chmod +x check_process.pl
```

Ce programme ne sera fonctionnel que si les modules Perl suivants sont installés sur le serveur :

```
cpan -i Nagios::Plugin Proc::ProcessTable
```

Vous pouvez tester le programme sur le serveur en ligne de commande :

Commande

```
root@SERVEURLINUX:~#
/usr/local/nagios/libexec/check_process.pl -n
mysqld
Check process OK - Process mysqld (3159) state
(sleep)
```

Il faut maintenant configurer NRPE sur le serveur Linux distant pour que ce programme puisse être interrogé à distance depuis « supervision ».

NRPE

```
# vi /usr/local/nagios/etc/nrpe.cfg
…
…
command[check_process]=/usr/local/nagios/libexec/
check_process.pl -n $ARG1$
```

L'argument « -n » est obligatoire pour notre programme. On ne tient pas compte de l'autre option.
Faisons un test en ligne de commande pour utiliser NRPE en local sur le serveur à superviser et depuis le serveur Nagios.

Depuis serveur Nagos

```
root@supervision:~#
/usr/local/nagios/libexec/check_nrpe -H
SERVERULINUX -c check_process -a mysqld
Check process OK - Process mysqld (3159) state
(sleep)
```

Tout fonctionne correctement !
Créons donc le service sur le serveur « supervision » pour que Nagios puisse surveiller le service MySQL !

Service check_uptime

```
define service{
    use                     generic-service
    host_name               SERVERULINUX
    service_description     Check MySQL
    check_command           check_nrpe!
check_process_ah!mysqld
    }
```

Retrouvez la suite de l'article de djibril en ligne : Lien 24

Un vent de migration : de nombreuses applications GTK passent à Qt

Ces derniers mois, trois applications GTK ainsi qu'un environnement de bureau sont passés au framework Qt :

- GCompris, une application éducative pour enfants, structurée en activités : Lien 25 ;
- Subsurface, une application de suivis de plongée, créée par Linus Torvalds : Lien 26 ;
- WireShark, logiciel d'analyse de paquets réseaux : Lien 27 ;
- LXDE, bureau X léger, qui collabore activement avec razor-qt en abandonnant GTK : Lien 28.

Ces trois applications ont un point commun : elles sont multiplateformes. C'est d'ailleurs là la plainte des développeurs contre GTK. En effet, ce framework possède des soucis de rendu sous Windows et OS X. De plus, le support des appareils mobiles (téléphones et tablettes) n'est pas simple à effectuer lors de l'utilisation de GTK.

Pour les développeurs de GCompris, le langage déclaratif Qt Quick est un outil appréciable pour la réalisation des interfaces et même de la logique de l'application (en JavaScript). Le manque de support des plateformes mobiles s'est aussi révélé compromettant pour la suite du projet.

Les développeurs de Subsurface – dont Linus Torvalds, qui n'hésite pourtant pas à crier son amour pour le C++ (Lien 29) – ont présenté une conférence à l'occasion du Linux.conf.au 2014, où Dirk Hohndel explique les raisons qui l'ont poussé à passer à Qt.

Du côté de Wireshark, le plus gros problème était l'intégration catastrophique de GTK avec l'interface d'OS X.

Bien entendu, ce changement ne se fait pas du jour au lendemain. Il est bien souvent nécessaire de recoder le cœur de l'application. Ce genre d'opération pourrait prendre des années, comme celle qui a mené Netscape à l'extinction : Lien 30.

Toutefois, chacun de ces projets avance à bon train, comme vous pourrez le constater sur leur dépôt. Ils considèrent tous ce changement comme bénéfique. GCompris proposera d'ailleurs des activités pour le prochain Google Summer of Code (Lien 31). Si le cœur vous en dit, n'hésitez plus à apprendre Qt ou Qt Quick et venir aider un projet open source !

Sources : Blog Wireshark (Lien 32), Blog LXDE (Lien 33), Gcompris (Lien 34)

Voir aussi : la vidéo de la conférence de Dirk Hohndel (Lien 35).

Commentez la news d'Alexandre Laurent en ligne : Lien 36

Sortie de Qt 5.2, avec la finalisation des ports pour iOS et Android, ainsi qu'une préversion pour Windows RT

Qt 5.2 vient de sortir. La précédente version, Qt 5.1, sortie en juillet, introduisait les ports pour Android et iOS et donnait un avant-goût de tout le travail effectué au niveau des capacités graphiques de Qt. Durant ces six derniers mois, de gros efforts ont été consentis afin de finaliser cette version et surtout ces nouveaux ports.

Qt pour mobiles est arrivé

Cette nouvelle version fait rentrer Qt dans la cour des grands, un *framework* permettant de développer pour la majorité des plates-formes mobiles : Android, iOS, BlackBerry, Sailfish/Jolla ainsi qu'Ubuntu mobile. Qt est le *framework* natif multiplate-forme qui couvre le plus large spectre de systèmes d'exploitation mobiles, spectre qui complète le nombre déjà important de plates-formes bureau et embarquées supportées. Qt 5.2 rend facile le déploiement de vos applications bureau ou embarqué pour les téléphones portables ou les tablettes.

Pour montrer l'engagement quant à la volonté d'être totalement multiplate-forme, voici un petit cadeau de Noël pour vous, une préversion de Qt pour WinRT (Lien 37). Cette préversion est basée sur la branche de développement et contient, de ce fait, de nouvelles fonctionnalités de Qt 5.3.

Qt sur Android et iOS

La plupart des API Qt sont supportés dans Qt 5.2 à la fois sous Android et sous iOS. Puisqu'il s'agit là de deux nouvelles plates-formes, il subsiste quelques exceptions. Qt WebKit (Lien 38) n'est pas encore totalement supporté sous Android et ne pourra pas être porté sous iOS à cause des restrictions techniques imposées par l'App Store. Néanmoins, les équipes travaillent dur afin de permettre l'utilisation de contenu web via les API Qt sur ces plates-formes. Pendant ce temps, vous êtes invités à utiliser les éléments natifs pour tout ce qui touche au contenu web. Les modules Qt Bluetooth (Lien 39) et Qt NFC (Lien 40) ne sont pas encore supportés mais seront implémentés dans les prochaines versions.
Toutes les autres API (y compris Qt Quick (Lien 41), Qt Sensors (Lien 42) et Qt Multimedia (Lien 43)) sont pleinement supportées sur ces plates-formes, permettant le développement d'applications très diverses uniquement avec les API Qt. Si quelque chose n'est pas encore supporté dans les API Qt, vous pouvez toujours vous

replier sur les API natives si besoin est. Pour Android, une API facilitant l'utilisation de la JNI (Java Native Interface) est fournie via le nouveau module Android Extras (Lien 44).

Le développement d'applications Qt sur mobile peut être intégralement fait via l'EDI Qt Creator pour Android, BlackBerry ainsi que pour Sailfish. Concernant iOS, son support par Qt Creator est encore expérimental.

À côté de ces nouvelles plates-formes mobiles, le développement sur les autres plates-formes reste très intensif. Plus de 1500 bogues ont été corrigés depuis la sortie de Qt 5.1.1. La plate-forme bureau a reçu beaucoup d'attention avec beaucoup d'améliorations concernant toutes les bibliothèques. De nouvelles fonctionnalités non-portables sont supportées.

De grandes améliorations pour le bureau

Les systèmes d'exploitation bureau sont au cœur même de Qt. Ainsi, beaucoup d'améliorations sont apportés par Qt 5.2 sur cette plate-forme :

- amélioration des contrôles Qt Quick (Lien 45) pour le bureau et facilitation de l'intégration de Qt Quick dans des applications basées sur Qwidget ;
- beaucoup de corrections de bogues et d'améliorations au niveau du module Qt Widgets ;
- ajout de la classe QkeySequenceEdit (Lien 46) facilitant la gestion des raccourcis clavier configurables par l'utilisateur ;
- l'accessibilité est pleinement supportée sur toutes les plates-formes bureau (ainsi qu'Android) : Lien 47 ;
- Qt Windows Extras module : intégration avec du code natif Windows (Lien 48) ;
- Qt Mac Extras module : intégration avec du code natif OS X (Lien 49) ;
- support des fuseaux horaires et des langues améliorés avec QTimeZone (Lien 50) et Qcollator (Lien 51) ;
- le Bluetooth est supporté sous Linux via le module Qt Bluetooth ;
- beaucoup de correctifs pour améliorer le support d'OS X Mavericks.

Tous ces changements font de Qt 5.2 une excellente base technique pour vos applications bureau.

Modernisation de Qt QML et de Qt Quick

Énormément de choses ont changé sous le capot. Le module Qt QML a reçu un tout nouveau moteur, permettant de supprimer la dépendance au moteur JavaScript V8. Le nouveau moteur, V4, a été conçu depuis zéro, en gardant en tête que QML sera le cas d'utilisation principal. Il supporte un mode interprété, ce qui lui permet de tourner sur des processeurs dont l'architecture ne permet pas le JIT ou bien sur des plates-formes où le JIT n'est pas autorisé, comme iOS où cette pratique est interdite par les règles de l'App Store. Dans les précédentes versions de Qt, l'intégration du moteur JavaScript V8 était difficile et causait des soucis de performances à l'interface entre JavaScript et Qt Quick. Ce problème est maintenant résolu grâce à ce nouveau moteur, qui fait directement usage des types de données Qt, ce qui permet une interaction avec du code Qt.

Comme bénéfice immédiat, vous constaterez des améliorations de performances dans la plupart des cas d'utilisation de QML. Néanmoins, étant donné que Qt 5.2 commence seulement à poser les pierres d'une nouvelle structure, les performances lors de l'exécution de beaucoup de logique en JavaScript peuvent être moins bonnes qu'avec Qt 5.1. Qt 5.2.1 apportera son lot d'améliorations de performances et les attentes dans ce domaine pour Qt 5.3 sont grandes.

Les choses ont aussi beaucoup changé du côté de Qt Quick : le moteur de rendu pour le nouveau graphe de scène a été complètement réécrit, ce qui lui permet d'être beaucoup plus performant pour le rendu (Lien 52) et de libérer plus de temps CPU pour l'application elle-même. De plus, chaque QQuickView effectue son rendu dans son propre fil d'exécution, ce qui permet d'être sûr que différentes scènes ne se bloqueront pas mutuellement.

Qt Quick se voit ajouter une nouvelle classe Animator (Lien 53), pour jouer des animations intégralement dans le fil d'exécution dédié au rendu. Ces animations ne peuvent pas se bloquer, même si le fil d'exécution principal est très chargé par de nombreux calculs.

Qt Creator 3.0 et autres

Qt 5.2 est livré avec le nouveau Qt Creator 3.0 (Lien 54). Cette nouvelle version améliore le support des plates-formes mobiles et améliore la stabilité des API concernant les plug-ins. Cela permet de créer une solide base pour des plug-ins tiers, ce vers quoi nous souhaiterions nous diriger dans le futur.

Différentes nouvelles API font leur apparition dans Qt 5.2. Le plus important est probablement le nouveau module Qt Positioning (Lien 55), le support du Bluetooth pour Linux et BlackBerry avec le module Qt Bluetooth (Lien 56), le support du NFC (Lien 57) pour BlackBerry, le support des fuseaux horaires, le support de la fusion des chaînes Unicode ainsi que les nouveaux modules additionnels pour Windows (Lien 58), OS X ainsi qu'Android (Lien 59).
Qt WebKit (Lien 60) a lui aussi reçu une mise à jour majeure, en se basant désormais sur une version de WebKit datant de cet été. Cela comprend beaucoup de nouvelles fonctionnalités telles que le CSS *blending*, la géolocalisation, les notifications web ainsi qu'un ramasse-miettes utilisant différents fils d'exécution.

Qt en action

Des applications Qt Quick ont été publiées sur les différents magasins d'applications mobiles afin que vous pussiez les tester. L'application « Quick Forecast » est une application de météo qui fait usage de l'API web et qui est écrite uniquement avec les contrôles de Qt Quick. Vous pouvez la télécharger pour Android depuis Google Play (Lien 61) ou pour iOS depuis l'App Store (Lien 62).

Téléchargez et testez

Vous pouvez télécharger Qt 5.2 depuis la page des téléchargements du Qt Project (Lien 63).

Commentez la news d'Arnold Dumas en ligne : Lien 64

Qt Creator 3.0 est disponible en version finale

Qt Creator 3.0 vient de sortir. En plus de nombreuses améliorations et de nouvelles fonctionnalités, cette nouvelle version améliore le support des différentes plates-formes. Celui d'Android est amélioré grâce à la nouvelle méthode de déploiement qui fait son arrivée avec Qt 5.2, méthode détaillée dans un article à part (Lien 65). Le NDK 10.2 de BlackBerry est maintenant supporté, comme un plus grand nombre d'appareils. Une nouvelle fonctionnalité encore expérimentale est le support d'iOS, ce qui vous permet de coder, compiler, déployer, exécuter et déboguer vos applications, sur le simulateur ou sur votre appareil, le tout sans quitter Qt Creator – mais en restant sous OS X. Le plug-in est désactivé par défaut mais, si vous installez Qt pour iOS via l'installateur de Qt 5.2, il sera alors automatiquement activé. Un support expérimental des appareils « quelconques » sur lesquels tourne un service compatible avec gdbserver/openocd est proposé.

Le support du C et du C++ a, une fois de plus, reçu de nombreuses améliorations. Notamment, les actions « Optimiser pour la boucle » et « Extraire la constante en tant que paramètre de fonction » font leur apparition ; lorsque vous cliquez sur « Suivre le symbole » sur un appel de fonction virtuelle, vous devrez maintenant décider vers quelle implémentation vous voulez aller.

De gros efforts ont été consentis afin de rendre le débogage avec LLDB pleinement utilisable. Les fonctionnalités proposées ne sont pas encore équivalentes à celles proposées pour gdb sous Linux mais sont en passe de surpasser celles de gdb sur OS X. Toutes les fonctionnalités principales de débogage sont présentes et beaucoup de plus avancées fonctionnent aussi, y compris une bonne partie des afficheurs de Qt ainsi que de la bibliothèque standard.

Le support du logiciel de gestion de versions Git a reçu de nombreuses améliorations. Il est maintenant possible de supprimer ou de renommer des préfixes, de choisir de suivre une branche distante, de cloner récursivement, de réserver ou de relâcher des parties de code directement depuis le visualiseur unifié de différences. Il est maintenant possible de passer de la vue côte-à-côte à la vue unifiée en appuyant sur un bouton du visualiseur de différences.

Bien sûr, énormément de choses ont changé et il n'est pas possible de lister tous les changements dans cet article. Vous devriez simplement télécharger cette nouvelle version et nous faire part de votre propre opinion.

Télécharger Qt Creator 3.0 : Lien 66

En lire plus sur Qt 5.2 : Lien 67

Commentez la news d'Arnold Dumas en ligne : Lien 68

Les derniers tutoriels et articles

Qt et OpenGL - Les tableaux de coordonnées

Cet article fait partie d'une série traitant de l'utilisation d'OpenGL avec Qt 5.1. Le premier article introduisait à l'utilisation d'OpenGL dans des applications Qt 5 : Lien 69

1. Les tableaux de coordonnées

Qt propose la classe QOpenGLBuffer (Lien 70) (anciennement nommée QGLBuffer (Lien 71)) pour faciliter la manipulation des différents types de tampons tels que les attributs de sommets ou les tampons d'indices. OpenGL possède aussi un conteneur appelé Vertex Array Object (VAO) pour faciliter la manipulation de ce type de données (Lien 72).
À partir de Qt 5.1, les VAO sont encapsulés dans la classe QopenGLVertexArrayObject (Lien 73). Lors de la liaison d'une instance de cette classe, OpenGL se « souvient » de la configuration des sommets déjà en place. Il sera plus tard possible de restaurer, très rapidement, l'état de ces sommets en se contentant de réactiver le VAO. Cela permet de rapidement changer l'état des sommets des objets à afficher :

```
void Scene::initialize()
{
    // Création d'un QOpenGLContext
    // m_shaderProgram est un QOpenGLShaderProgram

    // Création d'un VAO pour le premier objet à afficher
    m_vao1 = new QOpenGLVertexArrayObject( this );
    m_vao1->create();
    m_vao1->bind();

    // Initialisation des VBO et IBO (un QOpenGLBuffer sert de tampon pour les données,
    // spécification du format, etc.). Ils seront mémorisés par le VAO lié à ce moment-là
    m_positionBuffer.create();
    m_positionBuffer.setUsagePattern( QOpenGLBuffer::StreamDraw );
    m_positionBuffer.bind();
    m_positionBuffer.allocate( positionData,
                               vertexCount * 3 * sizeof( float ) );
    m_shaderProgram.enableAttributeArray( "vertexPosition" );
    m_shaderProgram.setAttributeBuffer( "vertexPosition", GL_FLOAT, 0, 3 );

    m_colorBuffer.create();
    m_colorBuffer.setUsagePattern( QOpenGLBuffer::StaticDraw );
```

```
    m_colorBuffer.bind();
    m_colorBuffer.allocate( colorData,
                            vertexCount * 3 *
sizeof( float ) );
    m_shaderProgram.enableAttributeArray( "vertex
Color" );
    m_shaderProgram.setAttributeBuffer( "vertexCo
lor", GL_FLOAT, 0, 3 );

    // Répétition des opérations pour les tampons
de normales, de coordonnées de textures, de
tangentes?
    ?

    // Crée le VAO du second objet
    m_vao2 = new
QOpenGLVertexArrayObject( this );
    m_vao2->create();
    m_vao2->bind();

    // Préparation des VBO et des IBO pour les
objets suivants
    ?

    // Nettoyage et on recommence pour les autres
objets
    m_skyBoxVAO = new
QOpenGLVertexArrayObject( this );
    ?
}

void Scene::render()
{
    // Vidage de la mémoire tampon
    m_funcs->glClear( GL_COLOR_BUFFER_BIT |
GL_DEPTH_BUFFER_BIT );

    // Liaison du shader, texture pour le premier
ensemble d'objets
    m_phongShaderProgram->bind();
    ?

    // Basculement sur les données des sommets
pour les premiers objets et rendu de ces derniers
    m_vao1->bind();
    m_funcs->glDrawElements(?);

    // Basculement sur les données du second
objet puis affichage de ce dernier
    m_vao2->bind();
    m_funcs->glDrawElements(?);

    // Changement éventuel du shader, des
textures ou d'autres objets?
    m_skyboxShaderProgram->bind();
    ?
    m_skyboxVAO->bind();
    m_funcs->glDrawElements(?);
    ?
}
```

Les VAO, introduits avec OpenGL 3, sont maintenant nécessaires pour OpenGL 3.1 et pour OpenGL >=3.2 avec le core profile. En plus de cela, les VAO sont disponibles au travers des extensions GL_ARB_vertex_array_object et via GL_OES_vertex_array_object pour OpenGL2 et OpenGL ES 2 respectivement. La classe QopenGLVertexArrayObject (Lien 74) utilisera les fonctions standard si disponibles et se rabattra sur l'extension appropriée, si possible, le cas échéant.
L'utilisation des VAO peut réellement simplifier le code de rendu et l'accélérer, étant donné que les pilotes graphiques ont moins de contrôles d'intégrité à faire. Cela s'explique par la réduction du nombre d'opérations sur les tampons.

Retrouvez l'article d'Arnold Dumas en ligne : Lien 75

Sortie de PyQt 5.2 : le binding de Riverbank poursuit son évolution et supporte dorénavant Qt 5.2

Riverbank publie aujourd'hui la nouvelle release de son binding Qt pour Python : PyQt 5.2
Cette version supporte intégralement Qt 5.2 et les nombreuses nouveautés que cette version du framework de Digia propose.

On notera notamment les nouveautés suivantes :

- nouvelle implémentation du graphe de scène Qt Quick, améliorant les performances de celui-ci : Lien 76 ;
- support complet d'Android et iOS : Lien 77.

Ces nouveautés incluent entre autre l'intégration des nouveaux modules QtBluetooth, QtPositioning, QtMacExtras, QtWinExtras et QtX11Extras.

Les efforts fournis par l'équipe Qt de Digia ayant permis d'obtenir un niveau de qualité important (Lien 78), Qt et PyQt se positionnent plus que jamais comme des outils performants pour le développement d'applications de tout types dont les applications mobiles.

Et vous ?
Utilisez-vous Qt ou son binding pour Python ?
Si oui, quelles sont les raisons qui ont motivé votre choix ?
Si non, que manque-t-il à ce framework et/ou à son binding pour sauter le pas ?

Commentez la news de Charlie Gentil en ligne : Lien 79

Java

Les dernières news Java

TLS 1.2 sera activé par défaut dans Java 8 - Le modèle mémoire du langage pourrait être révisé

La sécurité du module de communication de Java a longuement été critiquée par les développeurs ces dernières années. Cette nouvelle décision d'Oracle viendrait renforcer ladite sécurité. Elle sera activée dans le kit de développement Java 8 (JDK 8) par défaut. Elle permettra ainsi de crypter les communications sur Internet.

Les développeurs Oracle ont déclaré dans un billet de blog que « *TLS est conçu pour crypter les conversations entre deux parties et d'assurer que d'autres ne puissent ni lire ni modifier la conversation. Lorsqu'il est combiné avec le certificat d'autorisations, un bon niveau de confiance est établi, nous savons qui est à l'autre bout de la conversation et la conservation est protégée contre l'écoute ou la modification* ».

Pour rappel, TLS (Transport Level Security) est un protocole de sécurisation des échanges assez populaire sur Internet. TLS 1.2 a été introduit dans Java 7 en 2011, mais n'était pas activé par défaut. Cette version est déjà supportée par Windows 7, Internet Explorer 9 et versions suivantes, .NET et Chrome. Elle sera supportée par Firefox 27 à partir de février. Elle était activée par défaut sur les serveurs sockets, mais désactivée au niveau des clients.

Par ailleurs, la plateforme Java pourrait également bénéficier d'une révision importante de son modèle mémoire. Doug Lea, spécialiste de la programmation concurrente, a tout récemment publié ses premières propositions pour le JEP 188 (JDK Enhancement Proposal). Ces propositions se basent sur la programmation concurrente pour la prochaine version du JDK. Cette amélioration est une première depuis 2004.

Les propositions faites concernent principalement le modèle mémoire. Elle voudrait le démêler et l'adapter à celui du C11 et C++11 de même qu'une amélioration du support par les JVM des outils pour les architectures à accès mémoire non uniforme (NUMA) et une meilleure étude de la mémoire. Si les efforts aboutissent avec succès, assez de bugs pourraient être corrigés sur la plateforme. Le nouveau modèle peut être utilisé pour revoir le cahier de charges de Java et améliorer ses implémentations.

Oracle a annoncé qu'il publiera Java le 18 mars prochain, même si cette version contient encore des bugs. La Release Candidate du JDK 8 est disponible depuis le 23 janvier dernier.
Source : Blog Oracle (Lien 80), Fork Join (Lien 81)

Commentez la news de Francis Walter en ligne : Lien 82

Les derniers tutoriels et articles

Test d'IntelliJ IDEA v13 - Premières impressions sur la nouvelle version de l'éditeur IntelliJ IDEA de JetBrains

Cet article a pour but de montrer les nouveautés apportées par la 13e version d'IntelliJ IDEA de JetBrains, et de présenter quelques-unes de ses forces en tant qu'outil de développement.

1. Qu'est-ce qu'IntelliJ

Faisons en premier lieu un rapide tour de l'outil.

1.1. Introduction

Depuis janvier 2001, la société JetBrains (Lien 83) édite le logiciel IntelliJ IDEA (Lien 84). Il s'agit d'un **EDI** (ou **IDE** en anglais), à savoir un Environnement de Développement Intégré (Integrated Development Environment), autrement dit un ensemble d'outils destinés au développement logiciel. IDEA est ainsi à mettre au même niveau - toutes proportions gardées - d'Eclipse (Lien 85) ou encore de NetBeans (Lien 86).
Pour information, **IntelliJ** fait référence à la plateforme commune de JetBrains pour tous leurs outils de développement, **IDEA** étant l'EDI de développement Java. Il est donc plus juste de dire « je travaille sur IDEA » que « je travaille sur IntelliJ », bien que ce soit souvent la seconde phrase qui soit la plus courante.

1.2. Versions et prix

IDEA existe en deux versions : **Community** et **Ultimate.** Pour faire simple, la version **Community**, gratuite, est avant tout destinée au développement d'applications « lourdes » Java, Scala et Android. Dès qu'il s'agit de développer des applications web, il faut se tourner vers l'édition **Ultimate**. Son prix, pour une licence personnelle, est de 179 € (hors promotion ou prix de mise à jour).
Un comparatif complet des deux éditions est visible ici : Lien 87.
Vous pouvez télécharger l'une des deux versions sur la page dédiée : Lien 88. Notez au passage qu'IntelliJ IDEA est compatible Windows, Mac et Linux.

1.3. Principales fonctionnalités

Je ne vais pas détailler ici toutes les fonctionnalités d'IDEA, il me faudrait un livre entier pour cela. Je passe toutefois en revue les principales fonctionnalités et

langages, frameworks ou outils supportés.

1.3.1. Langages et framework supportés

La version **Community** gère nativement les langages suivants : Java, Scala, Groovy, Clojure et XML, XSD et DTD. Avec la version **Ultimate** s'ajoutent les langages dédiés au développement web, à savoir le HTML, CSS, JavaScript, CoffeeScript, ActionScript. Viennent également le support du Freemarker, de Velocity, du XSL, XPath, SQL, Ruby et JRuby, Python ou encore PHP.
Certains de ces langages nécessitent toutefois l'ajout de plugins gratuits.
Avec la version **Ultimate** vient également le support des frameworks les plus courants pour le développement autour de l'écosystème de la JVM ou du web. On citera par exemple Spring, Play! framework, JavaEE 6, GWT, Hibernate, Struts, Grails, Griffon, Sass, LESS, Rails, Django, Node.js, etc.

1.3.2. Gestionnaires de sources

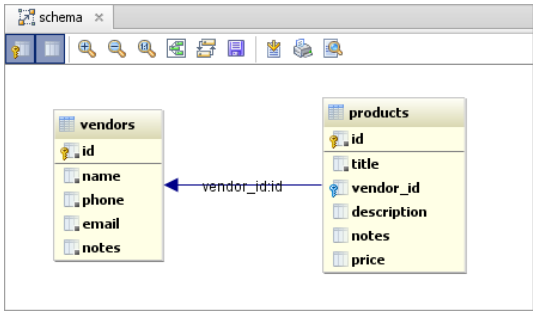
Si vous travaillez sur CVS (désolé pour vous), SVN, Mercurial ou Git (ou GitHub), alors la version **Community** sera suffisante pour vous. Si vous avez Team Foundation Server, Perforce, ClearCase ou encore Visual SourceSafe, c'est vers la version **Ultimate** qu'il vous faudra vous tourner.

1.3.3. Outils de construction

Les principaux outils de construction d'applications sont présents dans les deux éditions d'IDEA. On y retrouve ainsi Maven, Grandle, Ant et Gant Build Tools.

1.3.4. Développement et autres fonctions

Pour ce qui concerne le développement à proprement parler, IDEA offre une excellente intégration des outils de tests (JUnit, TestNG, Spock ou encore Cucumber), un historique local des modifications de fichiers, une interopérabilité avec Eclipse, ou encore un gestionnaire de contexte. Ce dernier permet de travailler sur un ticket JIRA (ou n'importe quel autre gestionnaire de tickets) et d'y associer un contexte. Ainsi, lorsque l'on rouvre un ticket JIRA sur lequel on avait déjà commencé à travailler, IDEA va rouvrir les fichiers qui étaient ouverts lorsque le contexte de ce ticket avait été précédemment fermé. Si vous êtes adeptes de Mylyn sur Eclipse (Lien 89), cette fonction devrait vous intéresser. Vous pouvez voir ici pour plus de détails (en anglais) : Lien 90.
À ce propos, IDEA s'interface sans problème avec les plus populaires des systèmes de tickets : JIRA, YouTrack, Lighthouse, GitHub, Redmine, Trac, etc.
Si vous possédez la version **Ultimate**, vous disposerez également d'un gestionnaire complet de base de données (éditeur SQL, définition de schémas, diagrammes, etc.), d'un outil de modélisation UML, d'outils de couverture de code, du « Structural Search and Replace » (voir le chapitre dédié plus loin).



Il est possible de générer le diagramme des tables grâce à IDEA

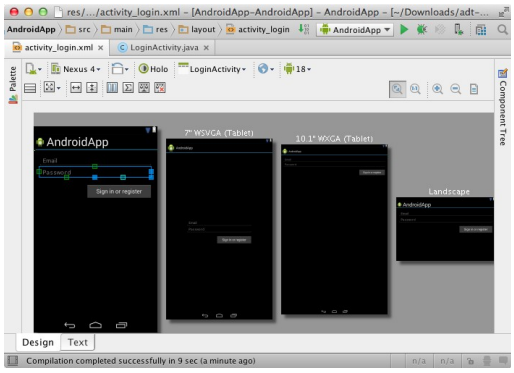
L'intégralité des fonctionnalités d'IDEA est à trouver sur cette page : Lien 91.

2. Les nouveautés de la version 13

Comme à chaque fin d'année, JetBrains publie sa nouvelle version majeure de son EDI. 2013 n'échappe pas à la règle, et a vu la version 13 de l'outil sortir. Quelles sont les principales nouveautés de cette version ? C'est ce que nous allons voir.

2.1. Android Studio

Les développeurs d'applications mobiles pour Android vont être contents. Annoncé lors du Google I/O de 2013 (Lien 92), IDEA est désormais l'EDI standard pour le développement d'applications mobiles pour l'OS de Google.



L'Android Studio

Cet Android Studio se compose de différents éléments :

- éditeur visuel pour la partie graphique des applications ;
- émulateur d'appareils Android ;
- meilleure intégration de Gradle (Lien 93), l'outil de construction des applications Android ;
- éditeur XML dédié à Android ;
- outil d'inspection Lint, pour analyser la qualité du code.

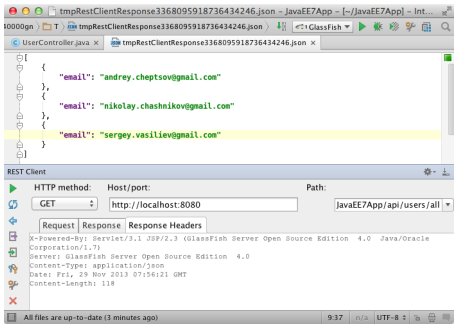
Autre bonne nouvelle, l'Android Studio est proposé dès la version **Community**.

2.2. Support de JavaEE 7 et Spring

Attention, ces nouveautés ne sont disponibles que pour la version **Ultimate**. La 13e édition d'IDEA dispose d'un support complet des spécifications JavaEE 7, ce qui implique :

- gestion du framework web Java Server Faces

(JSF) 2.2 ;
- support du Contexts and Dependency Injection (CDI) 1.1 ;
- introduction des nouvelles annotations de JAX-RS 2.0 pour les web services RESTful ;
- amélioration du client REST intégré ;
- présence des annotations utilisées pour les Web Sockets ;
- intégration des nouvelles versions des serveurs d'applications : Glassfish 4.0, WildFly 8, Tomcat 8, etc.

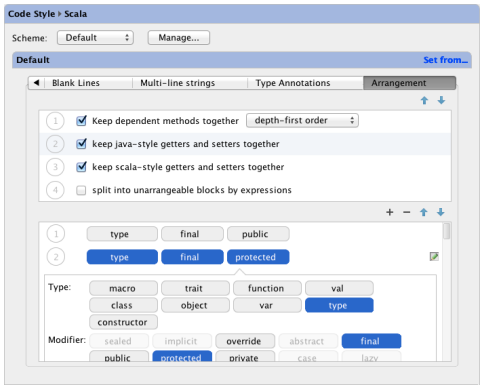


Le client REST intégré

Pour ce qui est de Spring, le support déjà bien présent dans les précédentes versions est encore amélioré. On y retrouve une meilleure gestion de Spring MVC permettant une navigation aisée entre les contrôleurs et les pages HTML associées, la détection des contextes non mappés et annotés avec @Context, ou encore des améliorations sur la performance, la documentation ou les fenêtres spécifiques à Spring (telle que celle de la vue des beans).

2.3. Langages alternatifs

Déjà bien supporté dans la précédente version, le support pour Scala ou Groovy s'améliore encore.
Pour Groovy, différentes fonctionnalités de refactoring ont été mises en place, comme l'introduction de variables ou de constantes. IDEA propose désormais la possibilité de créer des tests avec la bibliothèque Spock (Lien 94). Différents nouveaux inspecteurs ont également été introduits.
Concernant Scala, IDEA offre le support de l'outil sbt (Lien 95), des options de réorganisation du code ou encore la génération de méthodes de base (telles que hashCode ou equals).



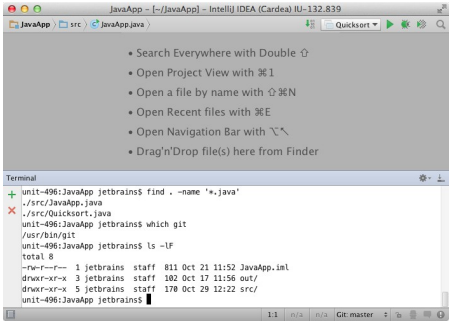
Le réorganisateur de code Scala

2.4. Développement web

Vous l'aurez deviné, les fonctionnalités propres au développement web ne sont disponibles que pour la version **Ultimate**. Les nouveautés sont entre autres :
- débogueur JavaScript amélioré, en particulier pour Google Chrome et Node.js : Lien 96 ;
- amélioration du refactoring CSS ;
- support de la bibliothèque de test Karma : Lien 97 ;
- support de la version 1.0 de Dart : Lien 98 ;
- début de l'intégration du futur standard des composants web : Lien 99 ;
- gestion des langages de templating comme Mustache (Lien 100) ou Handlebars (Lien 101).

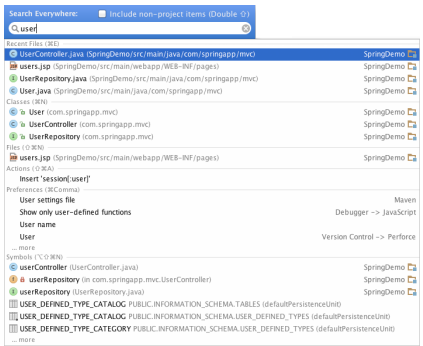
2.5. Autres fonctionnalités

Pour ce qui est des gestionnaires de sources, différentes améliorations sont apportées pour Git ou Mercurial. On notera ainsi l'ajout d'une fonctionnalité de « pull-request » pour GitHub.
Pour celles et ceux qui sont encore sur Subversion (SVN), ils seront heureux d'apprendre qu'IDEA est enfin compatible avec Subversion 1.8.
Concernant l'interface graphique, différentes améliorations ont été apportées, les thèmes Darcula et Light étant toujours présents. À noter la présence d'un mode « Présentation » qui ravira celles et ceux qui font des présentations techniques et qui doivent montrer du code.
Dans l'optique d'avoir tout sous la main dans un même outil, IDEA intègre désormais la possibilité d'exécuter des commandes shell localement ou à distance (via une connexion SSH) directement par la vue « Ligne de commandes ».



Un terminal complètement intégré à IDEA

Pour améliorer la productivité, la nouvelle fonction de recherche permet d'accéder à pratiquement tout à travers la même fenêtre : classe, méthode, action, paramétrage, etc.



IDEA propose des recherches vraiment complètes

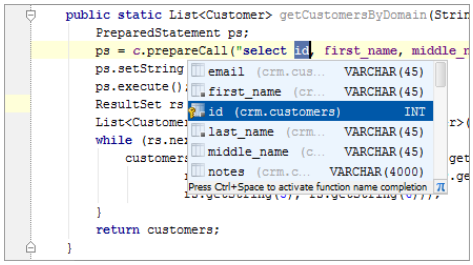
Vous pouvez retrouver toutes ces nouveautés, et bien plus encore sur la page dédiée du site de JetBrains : Lien 102.

3. Ce qui fait la force d'IntelliJ

On reproche souvent à IntelliJ IDEA d'être payant et d'avoir un prix relativement élevé. Une licence unique coûte 179 € si elle est personnelle, 449 € s'il s'agit d'une licence d'entreprise. Pour cette dernière, cela représente généralement moins d'une journée de prestation et cela risque de lui en faire gagner bien plus.
Ceux qui ont vraiment goûté à cet EDI, comme moi, n'ont plus envie de faire machine arrière et ne peuvent plus se passer de l'outil. Voyons dans ce chapitre quelques pistes permettant d'expliquer ceci.

3.1. L'autocomplétion

L'autocomplétion est une fonctionnalité absolument indispensable à tout bon outil de développement. IDEA non seulement n'échappe pas à la règle, mais propose sans aucun doute la plus puissante dans sa catégorie. Tout d'abord, la complétion « basique » qui consiste à aider le développeur à écrire son code Java fonctionne sans faille. Mais avec IDEA, elle va plus loin. Elle est capable d'aider à compléter le nom des variables ou des classes par exemple. Elle prend également en considération le contexte actuel, ne proposant ainsi que les types qui sont compatibles dans le contexte de la ligne de code courante. IntelliJ IDEA peut ainsi aider le développeur à compléter les requêtes HQL présentes dans des chaînes de caractères dans le code Java.



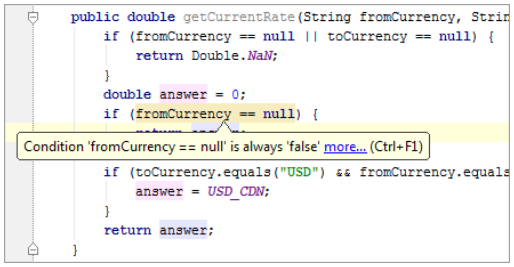
Une autocomplétion vraiment puissante

La puissance de l'autocomplétion ne se borne pas au langage Java, puisqu'il supporte sans broncher les langages web - HTML, CSS ou JavaScript -, le XML (il est par exemple capable de compléter les noms des classes Java dans les fichiers Spring), Scala, Groovy, etc. En fait, il n'y a pratiquement aucun endroit où IDEA n'est pas à même de proposer une autocomplétion efficace.
Si le sujet vous intéresse, vous pouvez lire l'article 20 code completions in IntelliJ IDEA (en anglais) (Lien 103), ou encore visualiser la vidéo disponible sur la page du site de JetBrains consacrée à l'autocomplétion : Lien 104.

3.2. Analyse et inspections

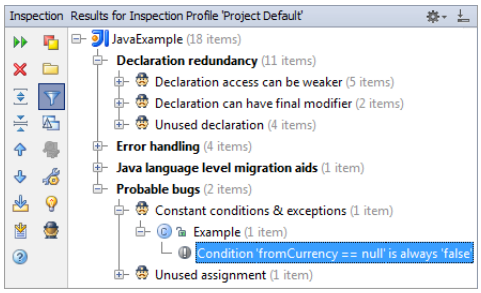
IntelliJ IDEA analyse en temps réel et en permanence votre code, à la recherche de problèmes potentiels. À chaque fois qu'il détecte une erreur, IDEA va surligner le problème, et proposera parfois même une solution. De même, lorsqu'IDEA détecte une inconsistance quelconque dans le code, il va proposer, grâce au principe des **inspections**, la meilleure correction possible. Ces inspections sont classées selon leur nature. Nous trouvons ainsi des inspections dédiées au style de codage, aux problèmes de performances ou de sécurité, d'autres dédiées à des langages précis (HTML, XML, Groovy, etc.) ou encore à des frameworks particuliers (Guice, GWT, Less, Maven, etc.).
Il existe plusieurs centaines d'inspections disponibles dans IDEA, difficile de les résumer ici, mais en voici quelques exemples :

- ajout d'un test de non-nullité, permettant d'éviter l'apparition de NullPointerException ;
- Une condition est toujours fausse (par exemple en testant la nullité d'un objet que l'on a initialisé plus tôt), mettant ainsi en avant du code mort ;
- possibilité de simplifier des expressions booléennes ;
- erreur dans la définition d'une entité Hibernate, par exemple des champs non présents dans la base de données ;
- code inutilisé : imports, variables globales, méthodes privées, propriété d'un fichier .properties inutilisée dans l'application, etc. ;
- copie manuelle de tableaux non optimisée ;
- dépendances dupliquées dans les pom.xml de Maven
- vérification de la validité des sélecteurs utilisés dans jQuery ;
- détection de problèmes dans la configuration des beans de Spring.



Un exemple de code potentiellement erroné détecté par IDEA

Il est toujours possible d'exécuter une inspection sur l'ensemble du projet, et parfois même de faire exécuter automatiquement la correction si celle-ci existe. Par exemple, je peux demander à IDEA tous les endroits où j'ai un size() == 0 qui pourrait être remplacé par un isEmpty(). IDEA va alors me lister tous les endroits où il trouve uneListe.size() == 0, mais aussi là où il va trouver uneListe.size() > 0 et me proposer de le remplacer par un uneListe.isEmpty() (ou !uneListe.isEmpty() dans le deuxième cas). Tout cela en un seul clic !



Il est possible d'exécuter toutes les inspections sur l'ensemble du projet

IDEA va devenir votre ami préféré pour écrire du joli code bien propre…

3.3. Outils de refactoring

Les fonctionnalités de refactoring proposées par IDEA en font également sa force. Il est ainsi possible de réaliser en un clic - ou une combinaison de touches - ce genre de choses :

- déplacement d'un morceau de code sélectionné vers une nouvelle méthode ;
- extraction d'une partie d'une classe vers une nouvelle, pour alléger la première ;
- inversion de conditions booléennes ;
- suppression sécurisée de composants. Lors de la suppression d'une méthode par exemple, IDEA va s'assurer qu'elle n'est plus utilisée dans le projet, et avertira le développeur le cas échéant.

Comme souvent, ces fonctionnalités puissantes ne se limitent pas au langage Java, et sont également présentes pour d'autres langages, comme le XML. On pourra ainsi renommer des nœuds ou des attributs, convertir des nœuds en attributs, etc.
Vous trouverez plus d'informations sur le refactoring sur cette page : Lien 105.

3.4. SSR, ou Structural Search and Replace

Le Structural Search and Replace (autrement dit la recherche et remplacement structurels) est un outil qui n'est pas toujours facile à manier, mais diablement efficace. Il n'est d'ailleurs disponible que dans la version **Ultimate**.
Il est parfois nécessaire de faire une recherche un peu complexe dans tout le code de son application. Cela peut-être une classe (class A implements B { }), une déclaration (a = b), un commentaire (// TODO A faire) ou une expression (new MaClasse();). Bref, des choses qu'une simple recherche n'est pas capable de trouver. Hé bien c'est le rôle du SSR. On ne lui donne pas un texte à chercher, mais plutôt un template.
Prenons un exemple : je souhaite utiliser la bibliothèque AssertJ (Lien 106) pour rendre mes assertions plus expressives dans mes tests unitaires. Avec JUnit, mon assertion s'écrit assertEquals(42, unEntier), alors qu'avec AssertJ, ce sera assertThat(unEntier).isEqualTo(42). Si je souhaite faire un changement automatique de toutes mes assertions en une seule fois, je vais avoir du mal avec une recherche normale. Avec le SSR, je peux faire cela en un clin d'œil. Tout d'abord, je saisis le template suivant :

Template pour le SSR
```
assertEquals($a$, $b$);
```

Vous noterez l'utilisation de variables a et b. Ensuite, je lui indique par quoi remplacer :

Template pour le SSR
```
assertThat($b$).isEqualTo($a$);
```

Et le tour est joué ! Pour être sûr de ne pas faire d'erreur, IDEA me montre une prévisualisation des changements avant de les appliquer.
Pour vous aider, il existe un certain nombre de templates prédéfinis, sur lesquels il sera plus prudent de partir pour arriver à ses fins.
Vous pouvez jeter un œil sur la documentation du SSR : Lien 107.

3.5. Petits conseils pour bien migrer vers IntelliJ IDEA

Fin 2012, j'ai acheté la version 12 d'IntelliJ, bien décidé à m'y mettre pour de vrai. J'avais déjà essayé par le passé d'y jeter un œil, mais mes habitudes sur Eclipse ont eu raison de toutes mes tentatives. Mais cette fois-ci fut la bonne, et je ne voudrais jamais avoir à faire machine arrière ! Pour vous aider à franchir le pas, voilà quelques petites astuces.
Tout d'abord, il faut persévérer un peu. C'est vrai qu'il est parfois déstabilisant de quitter un outil - en l'occurrence Eclipse - dans lequel on a pris ses habitudes au fil des ans, mais croyez-moi, cette fois c'est pour votre bien !
Premier point : la vue « Workspace » d'Eclipse disparait. IDEA ne gère qu'un seul projet à la fois par fenêtre (il est toutefois possible d'ouvrir autant de fenêtres d'IDEA que voulu). Finalement, ce n'est guère gênant, sauf si on a l'habitude de travailler sur dix projets en même temps (mais là, il y a sans doute un problème, non ?).
Autre chose : les raccourcis clavier. On ne peut pas travailler efficacement sur un outil sans en connaître les raccourcis clavier, du moins les principaux. IDEA propose une option pour faire correspondre autant que possible les raccourcis clavier à ceux d'Eclipse ou de NetBeans. Pour cela, il faut aller dans Settings > Keymap puis choisir Eclipse ou NetBeans dans le sélecteur Keymaps. Toutefois, je vous conseille vivement de laisser les raccourcis par défaut et de les apprendre. Pour vous aider, vous pouvez télécharger et imprimer un pense-bête (Lien 108) (la version pour Mac : Lien 109) ou pourquoi pas vous acheter un t-shirt (Lien 110)…

Le t-shirt IDEA avec les principaux raccourcis

Il existe aussi le raccourci « universel », Ctrl + Shift + A, qui vous permet d'exécuter n'importe quelle action en tapant simplement son nom. Enfin, je vous recommande le plugin « Key Promoter » qui détecte quand vous réalisez une action alors qu'un raccourci clavier existe. Dans pareille situation, le plugin va afficher une popup vous indiquant le raccourci à utiliser pour gagner du temps. C'est très pratique.
Si certaines personnes de votre équipe travaillent toujours sur Eclipse (les pauvres !), alors il faudra peut-être penser à ajouter le plugin de formatage d'Eclipse (Lien 111), ce qui vous assurera d'avoir les mêmes conventions que vos camarades. À noter aussi qu'il est possible de demander à IDEA de sauvegarder les métadonnées du projet au format Eclipse (fichier .classpath) plutôt qu'IDEA (fichier *.iml) :

Lien 112.
Quoi qu'il en soit, si vous vous sentez perdu, n'hésitez pas à consulter la F.A.Q. (Lien 113) ou le forum (Lien 114) dédié de Developpez.com.
D'autres conseils sur la page dédiée de JetBrains : Lien 115. Une page similaire existe si vous êtes un utilisateur de NetBeans : Lien 116.

4. Conclusion

Sans être une révolution, cette nouvelle version d'IntelliJ IDEA améliore encore un outil extrêmement bien fait et ultraproductif.

Si vous êtes développeur d'applications Android, alors il ne faut pas hésiter et se jeter (au moins) sur la version **Community**.
Si vous possédez déjà une version précédente de l'outil, il faudra sans doute fouiller un peu dans la liste des nouveautés pour voir si une mise à jour s'avère nécessaire.
Si vous n'avez encore jamais mis les mains sur cet outil, alors il ne faut vraiment pas manquer l'occasion d'y jeter un œil, soit avec la version **Community**, soit avec la version d'essai de 30 jours de la version **Ultimate.**

Retrouvez l'article de Romain Linsolas en ligne : Lien 117

La fondation Eclipse célèbre son dixième anniversaire, petit tour d'horizon sur le parcours de l'organisation

En 2001, IBM a rendu l'éditeur Eclipse open source. Quelques années plus tard, le 02 février 2004, s'est formée une organisation non-gouvernementale à but non lucratif qui s'est érigée en superviseur du développement de l'IDE et des projets gravitant autour; il s'agit bien sûr de la Fondation Eclipse.

En dix ans, la Fondation est passée de 50 à 205 membres et de 19 à 247 projets au total. D'ailleurs, à l'occasion des dix ans de l'IDE, l'un de ses projets consistait à la création d'un nouveau langage: Xtend. Un langage fortement inspiré de Java mais dont l'objectif était de «*fournir une alternative pertinente dans les situations où Java ne brille pas*».

Avec des fonctionnalités comme l'inférence de type, Xtend veut réduire la verbosité du langage tandis que d'autres, comme des instructions switch complexes, closures ou expressions templates sont venues pour améliorer son expressivité. Le projet se présente comme un ensemble de plug-ins qui se greffent à une installation existante de l'IDE.

Par la suite (2011) Eclipse s'est organisé en 11 « projets de haut niveau », chacun de ces projets pouvant en contenir de nombreux autres. Parmi eux figurait par exemple le Java Development Tools Project qui visait un environnement complet de développement pour le langage Java. Notons également le AJAX Toolkit Framework qui voulait apporter un outillage pour le développement d'applications AJAX. Mais aussi AspectJ pour la programmation orientée aspect pour Java dans Eclipse.

« *Dix ans c'est un long moment dans l'industrie de la technologie* » a commenté Mike Milinkovich, Directeur Exécutif de la Fondation. Rappelant combien tous les membres sont fiers de tout ce qui a été construit et gardent leur regard fixé en avant pour plus de succès dans le futur.

Source : Fondation Eclipse : Lien 118

Commentez la news de Stéphane Le Calme en ligne : Lien 119

Règles d'application des styles CSS et gestion des conflits - Comprendre les notions de cascade, d'héritage et de spécificité

Lequel d'entre nous ne s'est jamais torturé l'esprit pendant des heures en se demandant pourquoi telle règle CSS ne s'appliquait pas aux éléments ciblés par un sélecteur ? Ou pire, pourquoi cette règle s'appliquait à certains éléments et pas à d'autres ?
Je pense que c'est arrivé à tout le monde, surtout en période d'apprentissage.

Cet article va vous expliquer les trois principes permettant d'appliquer des styles à un élément : la cascade, l'héritage et la spécificité.

1. La cascade

La cascade correspond à la possibilité pour une balise de définir des styles dans plusieurs règles distinctes.
Au fur et à mesure que vous allez rajouter du contenu et des fonctionnalités à votre site, le besoin de mise en forme va augmenter.
Vous allez donc devoir créer des styles spécifiques pour différents éléments, mais parmi ces éléments, certains devront aussi se voir appliquer des styles liés à d'autres besoins.
Grâce à la cascade, un même élément peut recevoir des styles à partir de plusieurs règles différentes.
Par exemple, vous souhaitez que toutes les balises ayant la classe surround possèdent une bordure fine et arrondie. Vous créez donc la règle CSS :

```
.surround{
    border: 1px solid gray;
    border-radius: 25px;
    margin: 10px 40px;
    padding: 10px;
}
```

Vous intégrez ensuite sur votre site une notion de type de contenu et souhaitez ajouter une couleur de fond en fonction de ce type. Vous créez donc deux nouvelles classes :

```
.info{
    background-color: #DDEEFF;
}
.idee{
    background-color: #FFFFDD;
}
```

Vous pouvez donc donner à vos éléments plusieurs de ces classes, les règles CSS de chacune de ces classes seront appliquées aux éléments ciblés.

Code de l'exemple

```
<p class="surround info">Voici un message
d'information. Il contient les styles de
<code>.surround</code> et <code>.info</code>.</p>
<p class="surround idee">Voici un message d'idée.
Il contient les styles de <code>.surround</code>
et <code>.idee</code>.</p>
```

1.1. En cas de conflit

Il existe un risque de conflit entre plusieurs règles CSS, par exemple, si vous donnez à un élément à la fois la classe info et idee, alors la même règle sera définie deux fois avec deux valeurs différentes.
Dans ce cas, c'est la dernière règle définie dans le code CSS qui sera appliquée.

Code de l'exemple

```
<p class="surround idee info">Voici un message
créant un conflit. Il contient les styles de
<code>.surround</code>, <code>.idee</code> et
<code>.info</code>.</p>
```

Attention : on parle bien de la dernière règle définie dans les feuilles de style et non de la règle correspondant à la dernière classe précisée dans le code HTML.
Dans l'exemple précédent, la classe idee est précisée avant dans le code HTML de l'attribut class, mais c'est son style qui est appliqué, car il apparaît en dernier dans la balise <style>.

Notes.
Le moteur de rendu parse les feuilles de style de façon séquentielle, c'est-à-dire dans l'ordre où elles sont écrites dans le code.
Les styles dits inline, c'est-à-dire définis dans l'attribut HTML style, sont toujours prioritaires sur ceux définis dans les feuilles de style.
En revanche, il n'y a pas de notion de priorité entre les styles internes (balises <style>) et les styles externes (balises <link />). On peut assimiler les feuilles de style externes comme des balises <style> qui seraient présentes à cet emplacement et dont le contenu serait celui appelé par la balise <link />.

2. L'héritage

L'héritage correspond à la possibilité pour une balise de se voir appliquer les styles d'une balise ancêtre.

Rappel.
Les différentes normes HTML ont en commun la façon de dériver du XML, avec comme conséquence que deux balises ne peuvent pas se chevaucher : une balise ouverte à l'intérieur d'une autre doit être fermée avant la fermeture

de la balise dans laquelle elle se trouve. Concrètement, cela signifie que vous pouvez faire :

```
<balise1>
    <balise2>
    </balise2>
</balise1>
```

mais pas :

```
<balise1>
    <balise2>
</balise1>
    </balise2>
```

De cette règle stricte découle une représentation arborescente de la structure du document (appelée DOM) qui sera utilisée par CSS.
Dans cette structure, on distingue, par rapport à un élément référent :

- ses ancêtres ;
- son parent ;
- ses frères ;
- ses enfants ;
- ses descendants.

Notez qu'une balise ne peut avoir qu'une seule balise parent, mais elle peut avoir zéro, une ou plusieurs balises des autres types.

Attention : en CSS, la notion de frère (et les sélecteurs d'éléments frères associés + et ~) correspond en fait aux frères suivants (following siblings), c'est-à-dire les balises se trouvant au même niveau de l'arborescence (même balise parent) mais après elle dans le code HTML.

Ainsi, à quelques exceptions près (lister ces exceptions sort du cadre de cet article), les styles appliqués à une balise seront aussi appliqués à toutes ses balises descendantes à moins qu'une règle spécifique annule cet héritage.
Par exemple, vous souhaitez introduire dans votre page la notion de message d'erreur. Ces messages devront être écrits en rouge et en gras avec un fond rouge pâle.
Vous pouvez donc déclarer des règles spécifiques pour ce type de message :

```
.erreur{
    background-color: #FFBCC5;
    color: red;
    font-weight: bold;
}
```

Tous les éléments descendants de la balise sur laquelle cette classe est définie hériteront de ce style.

Code de l'exemple

```
<p class="surround erreur">Ceci est <i>un
message</i> d'<del>erreur</del> (<small>ne pas en
abuser</small>). Il contient des balises enfants
qui héritent de la classe
<code>.erreur</code>.</p>
```

Imaginons enfin que dans ce message, vous souhaitiez que la balise <code> soit écrite en noir normal avec un fond gris. Vous pouvez annuler l'héritage en définissant une classe pour cet élément :

```
.format-code{
    background-color: #F1F2F1;
    color: black;
    font-weight: normal;
}
```

Code de l'exemple

```
<p class="surround erreur">Ceci est <i>un
message</i> d'<del>erreur</del> (<small>ne pas en
abuser</small>). Il contient des balises enfants
qui héritent de la classe <code class="format-
code">.erreur</code>.</p>
```

Note.
Dans l'exemple précédent (ainsi que dans les autres), il aurait été possible d'affecter les styles directement aux balises <code>, mais dans le cadre de cet article, nous sommes obligés d'utiliser une classe pour éviter d'impacter les autres éléments qui n'ont pas de rapport avec les exemples.

Il n'est pas possible d'indiquer en CSS que l'on veut empêcher l'héritage. La seule façon de le faire est donc de redéfinir les styles qui ne doivent plus être hérités.
Même si cela peut sembler contraignant au premier abord, c'est néanmoins tout à fait logique : la mise en forme d'une page se doit de respecter une cohérence (notamment visuelle) que l'héritage rend simple à mettre en place.

2.1. En cas de conflit

On peut considérer qu'il y a conflit lorsque plusieurs éléments ancêtres définissent des valeurs différentes pour une même règle.
Dans ce cas, ce sont les styles de l'élément le plus proche dans l'arborescence qui sont appliqués.

Code de l'exemple

```
<div class="surround erreur">
    <div class="info">
        <div>Voici un message créant un conflit.
Il hérite à la fois de la classe <code
class="format-code">.erreur</code> et de la
classe <code class="format-
code">.info</code>.</div>
    </div>
</div>
```

Dans l'exemple ci-dessus, l'ancêtre .erreur va faire hériter tous ses descendants de ses styles.
Puis les styles du parent .info vont être appliqués.
On remarque donc que les styles du message héritent du parent (couleur de fond), mais que ceux qui n'ont pas été redéfinis (couleur et graisse du texte) restent appliqués.

Note.
On remarque dans l'exemple précédent que les propriétés liées aux bordures ne sont pas concernées par l'héritage. Ce qui est souhaitable, car si tous les éléments descendants héritaient d'une bordure fine arrondie, le texte deviendrait vite illisible !

3. La spécificité

La spécificité va permettre de déterminer le poids d'un sélecteur pour affecter les styles qui entrent en conflit.

C'est vrai qu'il a été dit (Lien 120) qu'en cas de conflit, c'est la dernière règle définie dans le code CSS qui est appliquée.
En réalité, ceci n'est appliqué que dans certains cas ultimes. Lorsqu'un même élément se voit attribuer plusieurs styles contradictoires, CSS va calculer le poids de chacun des sélecteurs ciblant cet élément.
C'est ensuite le style défini par le sélecteur ayant le poids le plus élevé (le plus spécifique) qui sera appliqué.

Note.
On entend ici par sélecteur l'ensemble des termes permettant de cibler un ou plusieurs éléments. Concrètement, il s'agit de tout ce qui précède l'accolade ouvrante.
Cependant, on parle aussi de sélecteur pour chacun des termes définissant un sélecteur composé.

3.1. Calcul du poids d'un sélecteur

Le poids d'un sélecteur est calculé en fonction de la nature et du nombre de chacun de ses termes. Il va permettre de déterminer trois nombres a, b et c :

- a est le nombre d'identifiants présents dans le sélecteur ;
- b est le nombre de classes, d'attributs ou de pseudoclasses ;
- c est le nombre d'éléments ou de pseudoéléments.

Comme il est rare de trouver plus de dix éléments dans un sélecteur, on peut ensuite se contenter de concaténer ces chiffres pour trouver le poids du sélecteur. Mais retenez tout de même que dix éléments de type classe ne valent pas un élément de type identifiant, ainsi un identifiant, 10 classes ou équivalents et 15 éléments donnent un poids de 1-10-15.
Une fois le poids de chaque sélecteur calculé, les styles appliqués seront ceux de celui ayant le poids le plus élevé.

Notes.
Le sélecteur universel (*) n'est pas pris en compte dans le calcul.
Les sélecteurs d'adjacence (~ et +) et de descendance (>) n'ont pas d'impact sur la spécificité.
La pseudoclasse de négation (:not()) n'est pas prise en compte pour le calcul, mais les éléments à l'intérieur de cette pseudoclasse le sont.

Attention : cette règle de la spécificité ne s'applique que pour les styles entrant en conflit.
Si un sélecteur a et un sélecteur b définissent différents styles pour une même balise, mais qu'un seul de ces styles est présent dans a et dans b, alors tous les autres styles seront appliqués et c'est le poids le plus élevé entre a et b qui déterminera la valeur du style commun.

Exemples.

Sélecteur	Poids	Explications
*	0	le sélecteur universel n'est pas compté
div	0-0-1	un élément
div span	0-0-2	deux éléments
div > ul::before	0-0-3	deux éléments et un pseudoélément
.classe	0-1-0	une classe
a[target=_blank]	0-1-1	un élément et un attribut
div .champ[disabled]	0-2-1	une classe, un attribut et un élément
#identifiant	1-0-0	un identifiant
#main .inner:empty	1-2-0	un identifiant, une classe et une pseudoclasse
#main :not(div)	1-0-1	la pseudoclasse :not() n'est pas comptée, mais son contenu l'est
#main .c1.c2.c3.c4.c5.c6.c7.c8.c9.c10	1-10-0	dix classes n'équivalent pas un identifiant

Note.
La spécificité des règles déclarées dans l'attribut HTML style sera toujours supérieure à celle des règles déclarées dans les feuilles de style.

Attention, le poids du sélecteur n'intervient jamais pour déterminer quelle règle sera héritée d'un ancêtre.
Concernant l'héritage, c'est toujours la règle définie par l'ancêtre le plus proche dans l'arborescence qui est appliquée.

3.2. En cas de conflit

Si deux sélecteurs ont un poids identique et définissent des valeurs différentes pour une règle, c'est alors la dernière règle définie dans le code qui est appliquée.

3.3. Exemple

Dans l'exemple suivant, nous utilisons un élément de classe .surround.
Nous créons différents sélecteurs correspondant à divers exemples qui vont permettre de permuter différentes combinaisons d'identifiants et de classes avec certains styles en conflit, d'autres non (il n'est pas possible de tester avec des sélecteurs de balise sous peine d'impacter l'ensemble de l'article).
Alternez les différentes combinaisons pour voir les répercussions sur l'affichage.
Voici les différents sélecteurs utilisés :

```
#identifiant{
/* Poids : 1-0-0 */
    background-color: #3A5486;
    box-shadow: 0 0 2px 5px #5096C8;
}
/* Doubler le sélecteur permet de renforcer son poids */
#identifiant2#identifiant2{
/* Poids : 2-0-0 */
    background-color: #5096C8;
    box-shadow: 0 0 2px 5px #3A5486;
}
.c2.c3.c4.c5.c6.c7.c8.c9.c10.c11.c12.c13.c14.c15{
/* Poids : 0-14-0 */
    background-color: #FFFFDD;
}
/* #identifiant2.c1 ne sera jamais appliqué à cause de #identifiant2#identifiant2 */
```

```
#identifiant.c1, #identifiant2.c1{
/* Poids : 1-1-0 */
    background-color: #DDEEFF;
    box-shadow: 0 0 2px 5px #5096C8 inset;
}
.c2{
/* Poids : 0-1-0 */
    background-color: #FFEEBB;
}
.c3{
/* Poids : 0-1-0 */
    background-color: #F1F2F1;
}
```

4. Conclusion

Savoir comment sont appliqués les styles notamment en cas de conflit est très important et permet de mieux organiser ses feuilles de style.
En général, il est recommandé de définir les styles des plus généraux aux plus particuliers sans utiliser de sélecteurs trop complexes.
N'oubliez pas non plus que les outils de développement intégrés à tous les navigateurs récents seront vos meilleurs amis pour déterminer quels styles sont appliqués à un élément ainsi que ceux qui ont été écrasés.

Retrouvez l'article de Didier Mouronval en ligne : Lien 121

Redécouvrez la mise en page avec Flexbox

Cet article va vous présenter les bases de l'utilisation des propriétés CSS3 flex.

Cet article est publié avec l'aimable autorisation de Synbioz, l'article original peut être lu sur le blog de Synbioz : Redécouvrez la mise en page avec Flexbox (Lien 122).

1. Qu'est qu'une boîte flexible ?

flexbox est un nouvel ensemble de propriétés CSS qui nous permet de gérer simplement la mise en page d'une série d'éléments au sein d'un élément parent et plus précisément de :

- mettre en place plusieurs éléments sur une ligne sans avoir à se soucier de la largeur de chacun d'eux ;
- modifier rapidement le sens de lecture vertical/horizontal ;
- aligner les éléments à gauche, à droite ou au centre du parent ;
- modifier l'ordre d'affichage des différents éléments ;
- déployer les éléments dans le parent sans problème pour gérer les marges ou la taille des éléments.

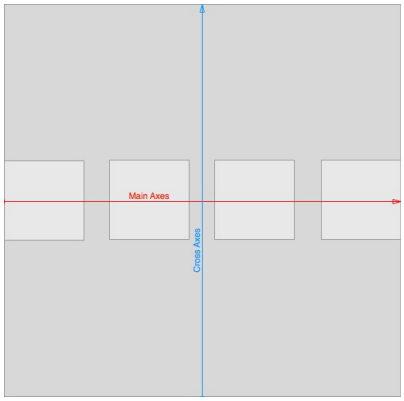
flexbox n'est pour le moment utilisable que sur les versions les plus récentes de nos navigateurs et avec des préfixes pour certains. Vous trouverez plus de précisions sur la compatibilité sur caniuse : Lien 123.
Pourquoi parler des flexbox si celles-ci ne sont pas encore utilisables sur tous les navigateurs ? Parce qu'il est toujours bon d'avoir un peu d'avance sur les technologies montantes, les flexbox sont beaucoup plus rapides à gérer qu'un tableau ou des éléments flottants pour maquetter un site dans le navigateur et parce qu'elles deviendront sans aucun doute un standard de mise en page.

1.1. Créons notre .flexcontainer

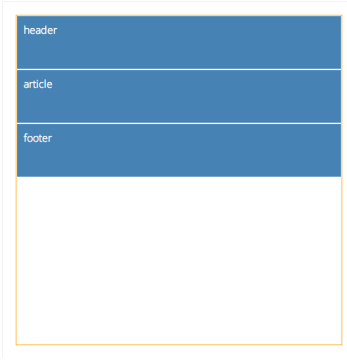
Tout d'abord il faut créer un conteneur pour nos éléments flex :

```
.flexcontainer {
    display: flex;
}
```

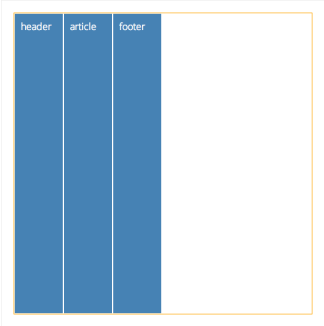
À savoir
Les flexbox utilisent un principe d'axes. Si on veut des éléments alignés verticalement, on utilise une colonne (column), pour des éléments horizontaux (par défaut) on utilise une ligne (row). Nous aurons donc deux axes appelés « Main axis », défini par le sens de lecture, et « Cross Axis » qui est perpendiculaire au premier.




```
.flexcontainer {
    display: flex;
    flex-flow: column;
    /* ou */
    flex-flow: row;
    /* flex-flow peut aussi s'écrire flex-
direction */
}
```



Column



Row

Pour définir l'ordre d'affichage des éléments directement depuis CSS, nous pouvons utiliser row-reverse ou column-reverse.

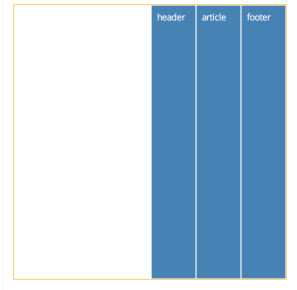
2. Utilisons les axes

Nous avons la possibilité de placer précisément les éléments dans notre flexbox, c'est ici que le principe d'axe vertical et horizontal prend son importance. Pour aligner les éléments à partir du début du main-axis, nous ajouterons la propriété justify-content: flex-start.

À retenir :

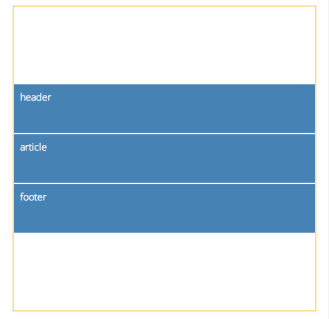
- justify-content : main-axis ;
- align-items : cross-axis.

```
.flexcontainer {
    flex-flow: row;
    align-items: flex-end;
}
```



Grâce à flexbox, nous avons enfin la possibilité de centrer un élément au sein d'un autre aussi bien horizontalement que verticalement.

```
.flexcontainer {
    flex-flow: column;
    justify-content: center;
}
```

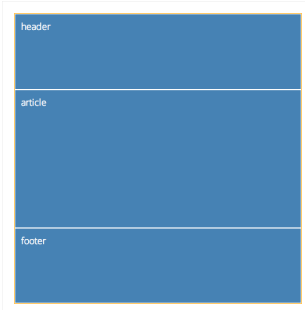


Aligner verticalement devient un jeu d'enfant !
Il est aussi possible de mettre des marges entre nos éléments, afin de remplir un espace donné ou simplement de séparer différents éléments. Pour cela, nous utilisons à nouveau la propriété justify-content avec comme valeur space-around. Les éléments de notre .flexcontainer seront séparés d'une marge identique et d'une demi-marge sur les cotés. La valeur space-between aura le même effet sans les marges extérieures aux éléments.
Pour le cross-axis, nous avons la possibilité de définir un alignement flex-end, flex-start, center ou stretch. Les deux premières valeurs positionnent les éléments sur les extrémités de l'axe, center place les éléments au centre du .flexcontainer et strech permet d'étirer les éléments afin qu'ils prennent la largeur ou hauteur du .flexcontainer.

3. Définir une taille

C'est bien pratique toutes ces nouvelles propriétés, mais j'ai souvent besoin d'éléments plus grands les uns que les autres. Voyons comment nous pouvons donner une taille à nos différents éléments dans la flexbox.

```
#first {
    flex: 1 ;
}
#second {
    flex: 2;
}
#third {
    flex: 1;
}
```



La première valeur correspond au flex-grow. Il n'y a pas d'unité : nous utilisons des proportions. Si ces trois éléments ont flex: 1;, alors ils feront tous la même taille. Si l'un de ces éléments a pour valeur 2, il prendra deux fois plus de place que les autres.
Il est aussi possible de définir une taille de base grâce à flex-basis. Lorsque le navigateur calculera la taille des éléments dans le .flexcontainer, l'espace restant après l'addition des différentes tailles des éléments sera divisé proportionnellement à la propriété flex-grow.
Il existe enfin la propriété flex-shrink qui fait la même chose que flex-grow mais pour réduire les éléments dans une flexbox trop petite.

À retenir
La propriété flex est un raccourci pour les propriétés :

- flex-grow ;
- flex-shrink ;
- flex-basis.

Soit flex: (grow) (shrink) (basis).

```
.element {
    flex: 1 1 100px;
}
```

4. Conclusion

Parmi les points les plus novateurs de l'amélioration du CSS ces dernières années, je retiendrais avant tout les boîtes flexibles. Elles nous permettront de nous passer de la plupart des systèmes de grille existants, qui sont souvent lourds, peu sémantiques et parfois encombrants. Une logique relativement simple, une gestion naturelle du responsive, des possibilités de mise en page bien plus avancées que celles que nous connaissions : les flexbox sont l'outil idéal pour construire vos pages directement dans le navigateur tout en vous laissant une grande flexibilité.

Retrouvez l'article de Synbioz en ligne : Lien 124

Enfin maîtriser les Expressions Rationnelles

Une couverture exhaustive des expressions rationnelles a jusqu'ici fait partie de l'atelier JS Puissant (Lien 125). Elle occupait bien 1 h, voire 1 h 30, en matinée.

C'est un sujet qui me tient véritablement à cœur, pour les raisons que je vais développer dans les deux premiers titres. Mais on m'a quelquefois fait remonter, très justement, qu'une telle couverture, au sein d'un atelier de 8 h, est tout à fait hors de proportion et que le temps libéré pourrait être utilement mis à profit en entrant plus dans le détail d'autres aspects (programmation fonctionnelle, structuration de code…).

Afin de ne pas simplement cesser de transmettre ces connaissances et de « convertir » les gens au bon usage des expressions rationnelles, j'ai donc opté pour l'approche inverse : sortir ce contenu des formations pour en faire un contenu librement accessible sur le site de JS Attitude !

Qui sait ? Peut-être cet article, s'il est suffisamment bien fichu, deviendra-t-il, à son petit niveau, une sorte de référence francophone sur ce sujet, comme le sont désormais des articles de Git Attitude (Lien 126) sur les clés SSH (Lien 127) ou l'installation de Gitosis (Lien 128)…

1. Le croque-mitaine

Traditionnellement, les expressions rationnelles ne sont pas enseignées. Il est déjà assez difficile de trouver un prof de BTS ou DUT capable de faire véritablement du Java, sans même parler de Python, Ruby ou JavaScript, pour espérer avoir carrément des cours décents d'expressions rationnelles.
Popularisées par Perl, les expressions rationnelles débarquent en général au travers d'un morceau de code parfaitement abscons, comme le dégorgement inattendu d'un fragment de fichier binaire au beau milieu du code source. Et de fait, quel développeur, pas forcément junior mais globalement sain d'esprit, n'aurait pas le cœur au bord des lèvres en tombant tout à coup sur ce genre de chose :

```
if (expr.match(/\w+\[([\w-]+)(?:[*~^$]?=(['"])?(.+?)\1\]/))
```

C'est vrai, quoi, comprenez-les. Les pauvres, personne ne les a prévenus, sinon ils n'auraient pas choisi le pâté de tripes à midi, vous pensez bien, c'était trop risqué.
Faute d'enseignement, faute de rationalité justement, faute de démystification utile, les expressions rationnelles restent à la périphérie de nos regards et de notre conscience, les banshees du développement, on les garde soigneusement à distance de peur qu'en y touchant elles ne nous attrapent tout entiers et nous volent ce qui nous reste de raison, tels de petits Cthulhus tout de symboles vêtus.

2. Le bon outil pour manipuler du texte

Naturellement, il n'en est rien, ne prêtez pas attention à cette porte qui claque quelque part dans l'asile d'Arkham. Les regex restent un outil extrêmement efficace pour extraire rapidement des éléments spécifiques d'un contenu textuel. Par exemple, repérer certaines balises dans du HTML ; trouver les lettres en double dans un texte ; capitaliser un texte (mettre ses initiales en majuscules et le reste en minuscules) ; détecter qu'un texte est bien une adresse e-mail (ou IP, ou URL…) valide ; etc.
Il est bien sûr possible d'écrire des algorithmes manuels pour tous ces usages, mais ceux-ci présentent deux inconvénients majeurs vis-à-vis des expressions rationnelles équivalentes :

- ils sont souvent largement plus verbeux (jusqu'à plusieurs centaines de lignes de code) et donc sujets à potentiellement bien davantage de bogues, notamment des erreurs de bornes (les erreurs +1/-1, ou fencepost errors) et de conditions de boucle ;
- ils sont en général significativement plus lents que leur équivalent à base d'expressions rationnelles.

Vous me rétorquerez peut-être qu'ils sont bien plus lisibles. C'est un argument aussi classique que fallacieux.
À part les cas triviaux d'analyse de texte (par exemple, en compter les consonnes), ce n'est pas vrai : tout traitement un tant soit peu avancé requiert automatiquement un volume non négligeable de code, avec ses variables, conditions, etc. La lisibilité se dégrade rapidement, ou à défaut, on obtient du code lisible, mais volumineux, dont il faut en outre comprendre l'algorithme pour en saisir l'objectif. Même si on se cantonne aux cas où le code alternatif est court, l'expression rationnelle équivalente est, elle, triviale et ne devrait donc pas poser de souci de

lisibilité.
Qui plus est, la lisibilité d'une expression rationnelle est perçue traditionnellement comme mauvaise pour deux raisons : d'une part, nombre de gens écrivent des expressions balourdes, pataudes, ne tirant pas le meilleur parti de la syntaxe qu'elles alourdissent inutilement. D'autre part, cette syntaxe, bien que grammaticalement assez simple, est effectivement assez peu intuitive : la majorité des développeurs ont donc connu le syndrome du « WTF?! » lors de leurs premières rencontres avec de telles expressions, ce qui entraîne un préjugé.
Enfin, nombre de personnes n'utilisent pas (où n'ont pas à disposition, comme c'est hélas le cas en JavaScript natif) le drapeau de syntaxe x, qui permet de découper et commenter une expression rationnelle lors de sa déclaration, ce qui accomplit des miracles en termes de lisibilité. Nous y reviendrons en fin d'article avec la bibliothèque XRegExp.

3. Un mot de vocabulaire

3.1. Expression rationnelle ou régulière ?

La terminologie anglaise ne connaît que regular expression, l'adjectif insistant sur le fait que, comme pour l'algèbre relationnelle, les expressions régulières sont basées sur des principes mathématiques déterministes et que la syntaxe est donc sans ambiguïté (si si !). En France, sans doute à la faveur de traductions initiales ayant pris quelques libertés, on s'est vite retrouvé avec les deux locutions **expression régulière** et **expression rationnelle**. On rencontre l'une à peu près aussi souvent que l'autre. J'ai personnellement pris le parti de la seconde, car je trouve que le **rationnelle** met davantage en avant ce principe de conception ; employez l'un ou l'autre comme bon vous semble, mais soyez juste cohérent avec vous-même !
Dans les deux cas, c'est tout de même assez long, aussi utilise-t-on la majeure partie du temps l'abréviation **regex**, en anglais comme en français, qui a en outre le mérite d'éviter le débat précédent. Dans le reste de cet article, j'emploierai cette abréviation dans un souci de concision.

3.2. « reguex » ou « redjex » ?

C'est la question suivante, posée par ceux qui se soucient de leur prononciation (et je les en félicite). Bien que l'origine du reg (regular) incite à l'utilisation d'un g dur (« reguex »), la prononciation anglaise usuelle ne s'intéresse qu'à l'abréviation elle-même et comme le g est doux devant les e et les i, on prononce effectivement « redjex ».

4. Les exemples classiques

Afin d'essayer de bien illustrer tout l'intérêt que peut représenter l'emploi des regex dans son code, en voici quelques-unes assez classiques. Le lecteur qui pinaille trouvera naturellement tout un tas de contre-exemples, mais le but n'est pas ici de recenser les solutions en béton armé, qui doivent gérer une telle pléthore de cas qu'elles sont en effet complexes (bien moins que les algorithmes associés, néanmoins). Il s'agit de montrer des cas pas trop ahurissants ! On n'aura par ailleurs recours qu'aux syntaxes gérées nativement par JavaScript (pas de groupes nommés, notamment).

- [0-9a-f]+ — Nombre entier hexadécimal.
- (["']).*?\1 — Chaîne de caractères (guillemets simples ou doubles, mais cohérents entre eux ; pas d'échappement).
- [\w.-]+@[\w-]+\.\w{3,6} — E-mail (basique).
- [\w-]+(\.[\w-]+){1,} — FQDN (nom de domaine, hors IRI).
- <[^>]+> — Balise XML (ouvrante ou fermante).
- \&(lt|gt|apos|quot|amp); — Échappement XML (les cinq entités obligatoires).
- (?:\d{1,3}\.){3}\d{1,3} — IP (basique).
- (?:(?:1?\d{1,2}|2[0-4]\d|25[0-5])\.){3}(?:1?\d{1,2}|2[0-4]\d|25[0-5]) — IP (exact : contrôle des valeurs).
- [a-z]+://[^:/]+(?::\d+)?(?:/[^?]+)?(?:\?[^#]+)?(?:#.+)? — URL (syntaxes des composants un peu laxistes…).

5. Déconstruire une expression rationnelle

Devant le foutoir (ben si, des fois, le foutoir !) que peut représenter une regex au premier abord, l'important est de savoir la déconstruire. Et là, c'est un peu comme les poupées russes : on procède de l'extérieur vers l'intérieur. La clé est aussi de savoir bien repérer les classes, groupes et leurs éventuelles imbrications, pour délimiter ainsi notre analyse.
Par exemple, prenons la première, [0-9a-f]+. On y repère une classe, c'est-à-dire une série de possibilités pour un unique caractère : [0-9a-f], qui est assez facile à décoder : de 0 à 9 et de a à f, donc un caractère hexadécimal. Et le quantificateur + vient indiquer qu'on s'attend à trouver ça au moins une fois.
Plus compliqué : (?:\d{1,3}\.){3}\d{1,3}. Là, on doit repérer tout de suite le groupe (non capturant en l'occurrence, mais passons) (?:\d{1,3}\.), et noter qu'il est quantifié 3 fois. On va même remarquer rapidement que la suite de l'expression est en fait une redite du contenu du groupe, au point final près. Les regex n'ont hélas pas de syntaxe pour dire « schéma X, N fois, séparé par schéma Y » : on se retrouve le plus souvent à dire « X + Y, N fois, puis X une dernière fois ». Sans pouvoir nommer les groupes, ça fait du doublon, comme ici. Et le contenu répété, \d{1,3}, est assez simple : « un chiffre décimal, 1 à 3 fois ».
Et ainsi de suite, en découpant le problème originel de plus en plus, pour se concentrer sur des fragments assez petits pour être analysés directement sans péter un câble. Mais croyez-moi, avec un peu de pratique, c'est beaucoup plus facile qu'il n'y paraît.

6. Construire une expression rationnelle

Pour construire une regex, on fait l'inverse : on part des petits composants, et on… compose, au fur et à mesure. Mieux on aura découpé à la base, plus la construction se fera naturellement, et moins on aura de répétition.
Supposons par exemple que vous souhaitiez créer une expression de validation d'un login, qui aurait les contraintes suivantes : uniquement des caractères alphanumériques ASCII, à raison d'un minimum de 6 et d'un maximum de 12. Comment s'y prend-on ?
Déjà, tous les caractères ont la même contrainte, ce qui est bon signe : ça va nous éviter de multiplier les composants. On doit donc d'abord exprimer la contrainte sur un caractère, sous forme d'une série de valeurs possibles.

C'est là le rôle des classes, on écrit donc [a-z0-9] pour lister les caractères et chiffres ASCII. Il ne nous reste plus qu'à quantifier ça, entre 6 et 12 occurrences : [a-z0-9]{6,12}.

6.1. Apprivoiser les expressions rationnelles pas à pas

La syntaxe des expressions rationnelles a finalement assez peu d'éléments potentiels. Et ce, d'autant plus que JavaScript ne gère pas nativement certaines extensions intéressantes, comme le drapeau x ou les groupes nommés. Nous allons examiner ces possibilités tranquillement, une par une, dans la suite de cet article.

6.2. Syntaxe des regex littérales en JavaScript

En JavaScript, vous avez deux manières d'obtenir une regex utilisable : en l'écrivant directement dans le code, de façon statique, sous forme d'un littéral. Ou en utilisant une construction d'objet explicite, avec la syntaxe new RegExp.
Dans la pratique, cette deuxième syntaxe n'a d'intérêt que si vous devez construire une regex dynamiquement sur la base de contenus que vous ne connaissez pas encore en écrivant le code (par exemple, une saisie utilisateur). Dans tous les autres cas, préférez la syntaxe littérale, plus concise.
Une regex littérale en JavaScript, ce sont trois choses : les délimiteurs, l'expression elle-même et les drapeaux de contrôle.

6.2.1. Délimiteurs

En JavaScript, on n'a pas le choix : une regex littérale est forcément encadrée par des slashes (/). Du coup, tout slash dans l'expression devra être échappé (\/), ce qui peut vite donner des trucs cocasses (/^https?:\/\//)…

6.2.2. Drapeaux (flags)

Les drapeaux de contrôle se placent après le slash de terminaison. JavaScript en reconnaît trois :

- i est très pratique et fréquent : Insensible à la casse. Permet d'éviter la galère de devoir spécifier tous nos caractères possibles en minuscules ET en majuscules, lorsqu'on accepte les deux…
- g rend la regex Globale : au lieu de se concentrer sur la première correspondance du texte, elle pourra agir sur toutes les correspondances. Change fondamentalement les comportements de recherche et de remplacement…
- m est souvent mal compris. Lui qui signifie Multilignes ne veut pas dire que notre regex va automatiquement marcher sur des correspondances à plusieurs lignes (et tout particulièrement, il ne permet pas au caractère générique total, ., de sauter les lignes), juste que les ancres (indications de position) ^ et $ correspondront aux bords de ligne et non aux bords du texte total. Ce drapeau sert plus rarement.

Ainsi par exemple, une regex de nombre hexadécimal sera plus probablement /[0-9a-f]/i, car dans la plupart des langages, la casse des lettres est sans importance pour les littéraux numériques.

6.3. Représenter un seul caractère de correspondance

Armés de ces connaissances de base sur la structure extérieure d'une regex, voyons l'intérieur. Une regex, fondamentalement, sert à chercher une correspondance dans un texte (voire à vérifier la correspondance du texte dans son ensemble). L'unité de base de ce travail, c'est le caractère. Quelles contraintes peut-on donc exprimer sur un seul caractère, pierre angulaire de la suite ?

6.3.1. Caractères littéraux (non spéciaux)

Tout d'abord, on peut exiger qu'il s'agisse d'un caractère précis. Lui et pas un autre. Il suffit alors d'utiliser le caractère directement, comme un littéral. Par exemple, a ne peut correspondre qu'au caractère A minuscule latin (code numérique 97).
Si votre caractère joue un rôle spécial dans les regex, il faudra l'échapper (sauf si vous êtes au sein d'une classe, ce qui la plupart du temps vous en dispense automatiquement, sympa). Par exemple, /?[a-z]=/i ne va pas faire correspondre un texte avec un point d'interrogation suivi d'une lettre ASCII suivie du signe égal : le point d'interrogation est un quantificateur, comme on le verra plus loin et s'attend donc même à avoir un contenu qui le précède : cette regex est invalide et votre interpréteur JS considérera ce morceau de code comme invalide. Il faudra employer /\?[a-z]=/i pour obtenir le résultat souhaité.

6.3.2. Classes explicites

Si on se contentait de comparer des caractères précis, on n'aurait pas besoin des regex : une bête comparaison de chaîne suffirait !
Afin d'exprimer une série de possibilités pour un caractère unique dans le texte analysé, on utilise des classes. Elles consistent à lister les caractères possibles, entre deux crochets. Par exemple, pour représenter les voyelles minuscules latines :

```
[aeiouy]
```

Évidemment, si on doit se farcir toutes les possibilités manuellement, ça va vite faire longuet… Lorsque les caractères possibles sont adjacents dans la table de caractères, on peut donc juste séparer les limites de la plage voulue par un tiret :

```
[a-z]
```

Et on peut naturellement combiner caractères explicites et plages, comme ci-dessous pour les chiffres hexadécimaux par exemple :

```
[a-f0-9]
```

Il peut, par ailleurs, arriver qu'on souhaite exprimer non pas tous les caractères autorisés, mais plutôt, par concision, tous les caractères interdits. On parle alors de classe négative, qui doit commencer par un circonflexe. Ainsi, pour interdire les chiffres arabes par exemple, mais autoriser tout le reste, on dirait ceci :

```
[^0-9]
```

Le sens spécial qu'ont le crochet fermant, le tiret et le circonflexe à l'intérieur d'une définition de classe oblige soit à les échapper (en les précédant d'un backslash), soit, pour le tiret, à le placer en bord de classe, afin de lever l'ambiguïté (s'il n'a pas de caractère des deux côtés, il n'indique pas un intervalle).

6.3.3. Classes prédéfinies

Certaines séries sont suffisamment communes pour disposer de syntaxes raccourcies. Celles-ci se présentent sous forme d'un backslash suivi d'une lettre. Les trois lettres gérées par JavaScript sont :

- d pour digit, qui regroupe les chiffres arabes ;
- w pour word, qui regroupe les lettres latines ASCII, les chiffres arabes et le tiret de soulignement (underscore) ;
- s pour space, qui regroupe les espacements usuels ASCII (espace, tabulations, saut de ligne, saut de page).

Si la lettre est minuscule, on a une classe positive (seuls ces caractères-là). Si elle est majuscule, on a une classe négative (tout sauf ces caractères-là). Ainsi, \d équivaut à [0-9] et \W équivaut à [^a-z0-9_].

6.3.4. ASCII vs. Unicode

Il est dommage que JavaScript ne gère pas mieux les jeux de caractères « réels », ceux employés par les langues. Ainsi, \w ignore les signes diacritiques (accents, cédilles, etc.) et d'une manière générale toutes les lettres non latines (grecques, allemandes, asiatiques…). Dans le même esprit, \s ignore l'espace insécable, aussi fréquente soit-elle.
Nous verrons en fin d'article qu'une petite bibliothèque JavaScript, XRegExp, étend considérablement les possibilités des regex en JavaScript, si vous avez ce type de besoin.

6.3.5. Combinaisons de classes

Il est possible de combiner des classes prédéfinies au sein d'une classe explicite, éventuellement avec d'autres caractères spécifiques, par exemple comme ceci :

```
[\wéèêëàâäîïôöûüç\s\u00a0]
```

On trouve ici les alphanumériques latin/arabe, le soulignement, les principales lettres françaises à signe diacritique, les espacements ASCII et l'espace insécable. JavaScript nous autorise en effet à utiliser la syntaxe de code littéral Unicode \uxxxx. On a ici le code 160 (a0 en hexadécimal), qui est l'espace insécable dans la table Unicode. C'est une syntaxe qu'on retrouve également dans CSS.

6.3.6. La classe générique : point

Une classe particulière est le Any char : . (point). Elle représente en théorie n'importe quel caractère. Dans la pratique, elle est limitée par l'activation (ou pas) d'un drapeau de comportement indiquant si elle comprend aussi les sauts de ligne ou non. Un tel drapeau n'existe pas dans les regex JavaScript standard et le point ne correspond jamais à un saut de ligne, même avec le mode multiligne (contrairement à d'autres moteurs de regex, comme par exemple celui de Ruby). Là encore, XRegExp offre une amélioration.
Dans tous les cas, il faut faire attention au risque d'employer un point littéral sans réfléchir, car le point des regex voudra dire « n'importe quel caractère » ! Pensez alors à l'échapper avec un backslash.

6.4. Représenter plusieurs caractères successifs

Jusqu'à présent, nous nous sommes concentrés sur l'établissement d'un critère pour un caractère. Mais dans une analyse de texte, c'est une séquence de caractères que nous allons examiner !

6.4.1. Parties fixes

Le plus simple consiste à utiliser, à la suite, les descriptifs de motif pour chaque caractère du texte à analyser. Par exemple, si vous voulez juste une lettre suivie d'un chiffre, vous direz sans doute :

```
[a-z]\d
```

Toutefois, la plupart du temps vous souhaitez aussi indiquer qu'un motif donné est acceptable pour davantage qu'un seul caractère : peut-être 2, 10, au moins 20, maximum 15, un ou plus, ou un nombre quelconque (y compris aucun)… C'est le rôle des quantificateurs.

6.4.2. Quantificateurs

Un quantificateur est une syntaxe spéciale qui, employée derrière un élément de la regex, modifie sa quantité potentielle. À la base, tout composant de la regex est attendu exactement une fois. On peut toutefois le faire suivre des syntaxes de quantification que voici :

- ? signifie « zéro ou un », ce qui revient à dire « optionnel » ;
- + signifie « au moins un » ;
- * signifie « un nombre quelconque de fois » (y compris zéro).

Il est aussi possible d'utiliser des bornes précises, au moyen d'une syntaxe un peu plus longue, encadrée par des accolades (curly braces) :

- {4} pour « exactement 4 fois » ;
- {4,12} pour « entre 4 et 12 fois » ;
- {4,} pour « au moins 4 fois ».

La syntaxe {,4} n'existe pas, pour la simple raison qu'on obtient le même résultat avec {0,4}.

6.4.3. Quantificateurs réticents

Les quantificateurs vus à l'instant sont tous qualifiés de gourmands (greedy quantifiers), au sens où ils tenteront systématiquement d'obtenir la plus grande correspondance possible. Ce n'est pas toujours ce qu'on veut. Prenons par exemple le texte analysé suivant :

```
<p><a href="#deuze">Et yop la deuze</a> - <a
href="#quarte">Et yop la quarte</a></p>
```

Imaginons maintenant que nous voulions récupérer les liens. Une regex naïve serait /<a.*>.+<\/a>/g. Mais ça va nous donner un unique résultat : "Philippe DUVAL2014-

02-20T21:38:46Ici, j'ai maintenu les guillemets droits, car je ne sais pas s'ils font partie de l'expression. S'ils n'en font pas partie, il faudra les remplacer par des guillemets français avec espace insécable à l'intérieur.Didier Mouronval2014-02-21T09:29:48.43Répondre à Philippe DUVAL (20/02/2014, 21:38): "..."Si, ils en font bien partie, ce qui n'est effectivement pas très visible dans l'odt mais l'est plus en ligne.Et yop la deuze - Et yop la quarte". Ah. Flûte. Le souci ici vient du fait qu'on utilise des quantificateurs gourmands. On pourrait bien sûr remplacer le premier par [^>]* histoire d'être sûrs de s'arrêter en fin de balise ouvrante, mais ça ne marche pas pour le deuxième, car si on faisait …>[^<]+, on s'arrêterait dès qu'un lien contient des balises…
On peut plutôt recourir à des versions réticentes.
Il suffit de rajouter un ? à la fin du quantificateur gourmand usuel :

```
/<a.*?>.+?<\/a>/g
```

Et là, on obtient bien deux résultats, un par lien. Les quantificateurs vont s'arrêter au premier chevron fermant rencontré et au premier rencontré.

6.5. Alternative entre deux séquences

Pour indiquer une alternative entre caractères, on utilise généralement une classe. Par exemple, [arz] signifie « a, ou r, ou z ». Mais comment faire pour des séquences ? On les sépare par un pipe (|).
Ainsi, hello|world signifie « hello ou world ». Et [1-9][0-9]*|0x[0-9a-f]+ signifie « un littéral nombre décimal (chiffre de 1 à 9 suivi éventuellement de chiffres décimaux) ou un littéral nombre hexadécimal (0 puis x puis un nombre quelconque non nul de chiffres hexadécimaux) ».
Attention cependant, il peut y avoir un piège quant à la portée de l'alternative : par défaut, elle s'applique à tout ce qui l'entoure ! C'est vite un souci, mais nous allons voir comment le résoudre dans un instant.

6.6. Groupes

Il est possible de rassembler des fragments de vos regex en groupes. Il y a plusieurs raisons de faire cela : pour quantifier plus qu'un simple caractère (mais carrément toute une séquence), pour limiter une alternative, ou pour obtenir comme résultat, en plus de la correspondance globale, des segments précis de celle-ci.
La syntaxe fondamentale des groupes est toujours la même : on entoure un groupe par des parenthèses : (et).

6.6.1. Grouper pour quantifier une séquence de caractères

Imaginons que vous souhaitiez faire une regex toute bête : « au moins deux mots ». Au sens « au moins deux séries de non-espacements », par exemple. La séquence « série de non-espacements » est assez facile : \S+ (majuscule). Et les mots seraient séparés par, justement, des séries d'espacements : \s+ (minuscule).
Pour dire « deux mots », on pourrait donc faire \S+\s+\S+. Hmmmm. Mais pour dire « au moins deux mots » ? On fait comment. Comme souvent lorsqu'il s'agit de séquences alternées (motifs X séparés par motif Y), on doit la représenter en deux parties :

- Motif X.
- Motif Y puis Motif X, le tout un certain nombre de fois (le tout quantifié, quoi).

Ici, vu qu'on veut « au moins deux mots », le quantificateur de cette deuxième partie serait « au moins une fois », donc le +. Mais on ne peut pas juste faire \S+\s+\S++, ça n'aurait aucun sens. Comment faire que ce dernier + sache à quoi il s'applique, en l'occurrence \s+\S+ ? Avec les groupes, pardi :

```
\S+(\s+\S+)+
```

Le groupe constitue une seule entité grammaticale, donc le quantificateur qui le suit s'applique au groupe dans son entier.

6.6.2. Grouper pour limiter une alternative

Un autre emploi fréquent du groupe est pour limiter une alternative (une utilisation de |, le pipe). Par défaut, on l'a vu, une alternative s'applique à tout ce qui l'entoure. Mais si elle figure au sein d'un groupe, elle se limite à ce groupe.
Supposons que vous vouliez autoriser les textes « hello world », « hallo world » et « hi world ». Le « world » étant commun on peut limiter l'alternative au premier mot. Mais si on fait ceci :

```
hello|hallo|hi world
```

…on se plante, car ça correspondrait à « hello », « hallo » ou « hi world », ce qui n'est pas ce qu'on veut. Il faut délimiter l'alternative avec un groupe :

```
(hello|hallo|hi) world
```

Et voilà, le tour est joué !

6.6.3. Grouper pour récupérer individuellement des fragments précis de la correspondance

Le dernier usage des groupes, qui est extrêmement utile, est la possibilité de référencer des portions spécifiques de la correspondance obtenue. Par exemple, vous essayez de repérer, au sein d'un texte, une chaîne française de date, au format jj/mm/aaaa. Non seulement vous voulez vérifier que la chaîne est bonne, mais aussi en extraire les composants. Au lieu de « juste » écrire ceci :

```
\d{2}/\d{2}/\d{4}
```

…vous pourriez isoler chaque composant dans son groupe :

```
(\d{2})/(\d{2})/(\d{4})
```

Vous allez alors obtenir, en plus de la correspondance globale (par exemple « 25/07/2012 ») des groupes 1 à 3 (« 25 », « 07 » et « 2012 »).
Les groupes sont numérotés de 1 à 9. Au-delà, vous ne disposez plus de numéros. Vous ne pouvez donc avoir recours « que » à 9 groupes numérotés dans une expression. On parle de groupes capturants.

6.6.4. Numérotation et imbrication de groupes

Il est naturellement possible d'imbriquer les groupes. Par exemple, comme ceci :

```
(\d{4}-(\d{2})-(\d{2}))
```

Mais alors, quel groupe a quel numéro, et quelle valeur ? La règle de numérotation est simple : dans l'ordre des parenthèses ouvrantes. Ainsi, sur le texte 2012-07-28, on obtiendrait :

- le groupe 1 : 2012-07-28 ;
- le groupe 2 : 07 ;
- et le groupe 3 : 28.

6.6.5. Références arrières (« backrefs »)

Dès l'instant où un groupe est numéroté, il devient possible de le référencer, tant pour un remplacement que pour une recherche.
Le cas de figure du remplacement est assez classique et utilise la syntaxe $numéro. Supposons que vous ayez à transformer des dates américaines (mois/jour/année) en dates techniques (année-mois-jour). On pourrait procéder ainsi :

```
usDate.replace(/(\d{2})\/(\d{2})\/(\d{4})/, '$3-$2-$1')
```

Là où ça devient vraiment pratique, c'est au sein même du motif de recherche. On s'en sert couramment pour faire correspondre des délimiteurs. Supposons que vous vouliez définir un motif qui autorise de part et d'autre l'apostrophe (') ou le guillemet ("), mais qui exige que les deux extrémités correspondent. La version pourrie, ce serait :

```
'.+?'|".+?"
```

Et encore, elle pose plein de soucis. Mais vous pouvez facilement simplifier ça (surtout si ce qu'il y a entre les apostrophes/guillemets est un motif compliqué) en évitant toute répétition. Pour le cas où l'apostrophe/guillemet initial constituerait le premier groupe numéroté du motif :

```
(['"]).+?\1
```

Pratique, non ? Le \1 fait référence au texte qui a correspondu au groupe 1 : il faut alors que dans le texte analysé, on trouve à l'endroit du \1 exactement le même texte que pour le groupe 1.

6.6.6. Groupes non capturants

Neuf groupes, c'est beaucoup et vous n'aurez normalement pas besoin de tous. En revanche, il arrive souvent que vous ne souhaitiez pas exploiter le groupe individuellement, et n'ayez donc pas besoin de lui attribuer un numéro. Par exemple, vous l'auriez utilisé uniquement pour une des autres raisons : quantifier une séquence ou limiter une alternative. Il est alors dommage de « polluer » les groupes numérotés légitimes avec ceux-là, vous obligeant à examiner les groupes 2, 6 et 9 au lieu de 1, 2 et 3. Comment faire ?
C'est tout simple, il suffit de recourir alors à des groupes non capturants. Par défaut, un groupe est capturant : il s'octroie un numéro. Mais lorsque vous l'utilisez juste à des fins structurelles, vous pouvez le rendre non capturant en préfixant son contenu par ?:. Ça donnerait quelque chose comme ça :

```
([A-Z]{2})(?:\.\d+)+([A-Z])
```

Dans l'exemple ci-dessus, on a d'abord un groupe capturant, qui comprend deux lettres latines majuscules : ([A-Z]{2}). C'est le groupe numéro 1. Puis on a un groupe non capturant, dont le seul objectif est de pouvoir ensuite appliquer à son contenu le quantificateur + : (?:\.\d+), à savoir un point suivi d'au moins un chiffre. Et finalement, on a un deuxième groupe capturant, qui aura donc le numéro 2 : ([A-Z]), soit une lettre latine majuscule.
En utilisant judicieusement la syntaxe non capturante, vous n'utilisez des numéros que pour les groupes que vous souhaitez effectivement référencer ensuite. En plus, un groupe capturant est souvent plus lourd à gérer par les moteurs de regex qu'un groupe non capturant : vous améliorez donc aussi, même si ce n'est pas toujours sensible, la performance de votre regex.

6.7. Indiquer la position avec les ancres

Jusqu'ici, nous avons rédigé des regex qui, dans la pratique, pouvaient correspondre à un texte figurant n'importe où dans la chaîne de caractères examinée.
Par exemple, la regex h[ea]ll?o correspondra bien sûr à hallo, hello et halo, mais aussi à hello world, say hello, mark! et surtout à chaloupe ou halogène.
Il arrive fréquemment qu'on ait besoin de préciser une contrainte de positionnement sur tout ou partie d'une regex. C'est le rôle des ancres.
Les regex JavaScript reconnaissent les ancres suivantes :

- ^ (circonflexe, en anglais caret). Il s'agit du début de la chaîne de caractère examinée. Si la regex utilise le drapeau multilignes (m), il peut en fait s'agir de n'importe quel début de ligne (début de texte ou juste après un saut de ligne, \n) ;
- $ (dollar). C'est la fin de la chaîne, ou en mode multiligne ce peut aussi être une fin de ligne ;
- \b est le word Boundary. C'est une ancre extrêmement utile, qui désigne une bordure de mot. Ça inclut notamment les sauts de lignes, espaces, la ponctuation et les extrémités du texte. Par symétrie, on a aussi \B (majuscule), qui comme pour les classes est le contraire de \b : elle indique qu'on ne se situe pas à la bordure d'un mot.

Quelques exemples :

```
^halo // correspond à "halo" et "halogène" et "halo éclatant", mais pas à "chaloupe" ou "un halo."
halo$ // correspond à "halo", mais ni "halogène", ni "halo éclatant", ni "chaloupe", ni "un halo."
\bhalo // correspond à "halo", "un halo.", "halo éclatant" et "halogène", mais pas "chaloupe"
halo\b // correspond à "halo", "un halo." et "halo éclatant", mais ni "halogène" ni "chaloupe"
```

Ainsi, pour indiquer qu'une regex correspond à l'ensemble d'un texte, on l'encadre par ^ et $, comme ceci :

```
^h[ea]llo w(?:orld|elt)$ // "hello world", "hallo
```

```
welt", "hello welt", "hallo world" et c'est tout.
```

6.8. Conditions sur la suite du texte (lookaheads)

Parfois, une contrainte sur la position ne suffit pas. Il faut carrément indiquer une condition sur la suite du texte.

6.8.1. Lookaheads positifs

Peut-être quelque chose comme « un chiffre, à condition qu'il soit suivi par deux autres chiffres ». Mais on ne veut pas pour autant accumuler les deux chiffres qui suivent dans la correspondance. On utilise pour cela un positive lookahead, identifié comme un groupe avec un préfixe ?= :

```
\d(?=\d\d) // "2" dans "237" ou "243", mais pas
dans "24M", "2", "2TZ" ou "2T4".
```

6.8.2. Lookaheads négatifs

Il est évidemment possible aussi d'exprimer une contrainte négative sur la suite du texte : ce qu'on ne veut pas trouver après. Au lieu de ?=, on utilise alors ?!. Pour prendre l'inverse de l'exemple précédent, on pourrait dire « un chiffre, pourvu qu'il ne soit pas suivi par un autre chiffre » :

```
\d(?!\d) // "2" dans "2", "2TZ" ou "2T4", mais
pas dans "237", "243" ou "24M".
```

Tout ça peut sembler un peu basique, mais voici un bel exemple combiné et imbriqué, et une fois que vous l'avez digéré, vous êtes au point sur les lookaheads ! Examinez plutôt :

```
numberString.replace(/(\d)(?=(\d\d\d)+(?!\d))/g,
"$1 ")
```

Oh pétard ! Ça fait quoi, ça ?!?
Eh bien « tout simplement », ça ajoute les espaces en séparateurs de milliers entre les groupes de trois chiffres (en partant de la droite, évidemment). En somme, grâce aux lookaheads, on arrive à analyser par la droite alors que les regex travaillent par la gauche.
Voilà de quoi vous faire tourner les méninges… Petite astuce pour décrypter cette regex : on cherche en fait à remplacer chaque fin de groupe numérique (hors le dernier) par son dernier chiffre suivi du séparateur. Et comment sait-on qu'on a des groupes numériques valables derrière soi ? Parce qu'on a un nombre de chiffres multiple de 3 derrière soi. Voilà, je vous laisse décortiquer tout ça…
Quelques exemples de conversion :

```
'243'          // '243'
'2431'         // '2 431'
'24315'        // '24 315'
'243157'       // '243 157'
'2431576'      // '2 431 576'
…
```

6.8.3. Lookbehinds ?

Dans certains cas de figure, il peut être également utile de poser une contrainte sur le texte d'avant, indépendamment de comment on a éventuellement traversé ce texte dans un début de correspondance. On aurait alors recours aux lookbehinds, qui traditionnellement s'écrivent ?<= et ?<!. Cependant, JavaScript ne propose pas, en standard, cette syntaxe.
Et pour le coup, XRegExp ne permet pas de simuler cette possibilité. Il existe toutefois la possibilité de les simuler si besoin (Lien 129).

Retrouvez la suite de l'article de Christophe Porteneuve en ligne : Lien 130

Introduction aux Styles FireMonkey XE4 - Mes premiers pas

Les contrôles VCL sont dessinés par l'OS alors que les contrôles le sont entièrement par FireMonkey lui-même. Cela veut dire que FireMonkey, et par là même votre application, a un contrôle total sur chaque rendu de votre application. Les styles permettent de changer radicalement l'apparence de votre application ou d'un contrôle particulier.

1. Introduction

1.1. Préambule

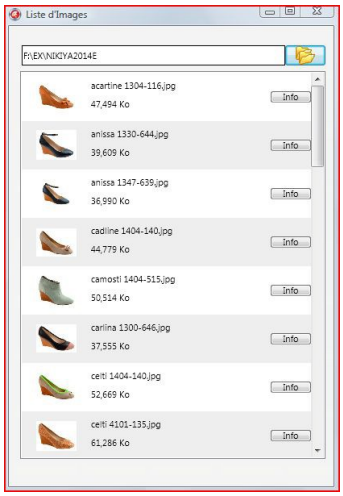
Ma première approche des styles a été douloureuse. J'ai tellement galéré sur les Styles FMX pour une application « simple » que je voudrais éviter aux autres, en particulier aux membres de www.developpez.net, d'avoir à faire les mêmes recherches que moi (longues heures de vidéos anglaises, « googlisages » intensifs, ouverture des programmes exemples, etc.). Voici un lien regroupant une partie de mes lectures et visionnages : Lien 131.

1.2. Mise en garde

Pour les « vieux delphistes », l'éditeur de style est déroutant. En effet, on a tendance à vouloir utiliser l'écran de design pour poser nos formes de base, et attendre un rendu à la mesure des designs d'une forme classique. Las, nous devons vite déchanter, les composants doivent être « posés » dans l'arbre de structure et le rafraîchissement du rendu n'est pas toujours instantané.
Mes premiers conseils : sauvegardez souvent, nommez (propriété **StyleName**) tous les éléments et ayez beaucoup de patience !

2. Objectif

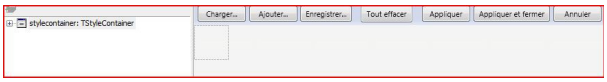
Mon objectif est de faire, sous Windows, la liste des images JPG d'un répertoire donné.



3. Pour comprendre : un label avec bouton

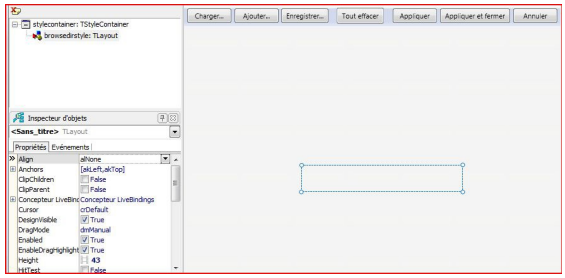
Pour commencer, et surtout essayer de comprendre, nous allons créer un style pour la recherche de répertoire. Tout d'abord, poser un **TSyleBook** sur la forme, et un **TLabel** (cela aurait pu être un **TEdit** mais je voulais faire simple). Il aurait été facile de faire la même chose directement et sans style en posant par exemple dans un **Layout** : un **TRectangle**, un **TLabel**, un **TButton**, un **TImage** comme sur la droite de l'image.

De fait, nous allons faire la même chose avec l'éditeur de style. Double-cliquez sur le **TStyleBook** que j'ai nommé **Resource1** et posez, dans l'arbre de structure, un premier **TLayout** et là : première surprise, on ne voit rien !



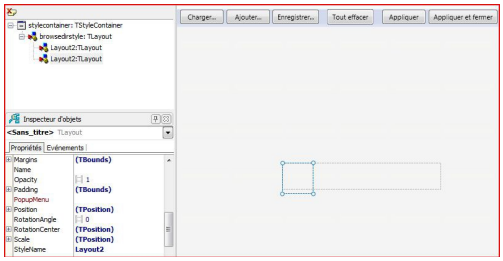
Pas de panique, ouvrez l'arborescence, cliquez sur le **Layout**, l'écran de design se rafraîchit et on peut le déplacer et le retailler à souhait. Profitons-en pour rajouter dans ses propriétés son **StyleName** (dans mon exemple : **BrowseDirStyle**).

Cliquez sur **Appliquer** et **Fermer** puis ré-ouvrez le style pour voir le résultat.

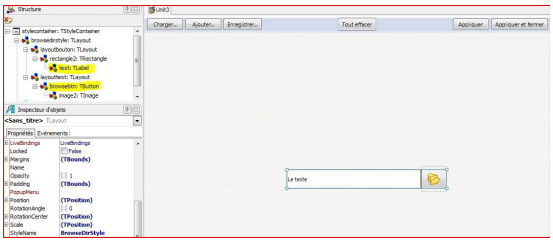


Ça y est, le nom est enfin pris en compte, mais si la taille du **Layout** est gardée, il est de nouveau centré dans la fenêtre de design. On n'en a pas fini avec les surprises de ce genre et il faudra faire de nombreuses fois la manipulation « Appliquer et Fermer + Ré-Ouvrir », que je désignerai maintenant sous le nom de **A.F.R.O**.
Déposons ensuite dans ce **Layout** deux autres **Layouts**, un qui sera cadré à droite, dans lequel on ajoutera un bouton (**TButton**), qui contiendra à son tour une image et l'autre

aligné à la zone client, qui contiendra un **TRectangle** contenant lui-même un **TLabel** pour finaliser l'ensemble.



Et voilà, nous sommes tombés dans un nouveau piège. (cf. mes premiers conseils).
Impossible de se positionner sur le premier **Layout**, car ils ont le même nom ! Nommez celui qui est accessible (**StyleName**), faites la manipulation **A.F.R.O** (qui va devenir célèbre), et recommencez pour le second **Layout**.
Sortis de ce piège, il ne nous reste plus qu'à finaliser l'ensemble. Ensuite, on va nommer (au minimum avec la propriété **StyleName**) les deux composants que nous allons utiliser (le label et le bouton) pour pouvoir y accéder. Pour rafraîchir le design, n'hésitez pas à cliquer sur un des parents du composant que vous venez d'ajouter (cf. mise en garde).



Vous remarquerez que j'ai nommé (**StyleName**) le **TLabel** 'Text', une petite astuce qui va permettre d'accéder à cette propriété directement.

Il nous reste maintenant à appliquer ça à notre forme et notre code. Posons un **TLabel** sur la forme et mettons ses propriétés **StyleName=Resources** et **StyleLookup=BrowsedirStyle** et la magie opère.
Pour changer l'intitulé du label, il suffira d'utiliser le code suivant :

```
MonLabelStylé.Text := 'Sélectionnez un
répertoire';
```

Malheureusement, l'astuce pour le label ne fonctionnera pas pour le bouton et nous devrons lier l'événement par code :

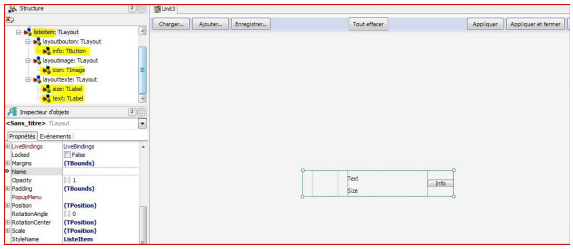
```
MonLabelStylé.StylesData[&#8216;BrowseBtn.OnClick
'] := TValue.From<TNotifyEvent>(Button3Click);
```

où **Button3Click** est une procédure à déclarer et bien sûr à coder.

4. Personnaliser les items d'une liste

Passons maintenant au style des items de la liste. Chaque item va contenir : une image, deux textes (un pour le nom de l'image, l'autre pour sa taille) et enfin un bouton qui nous permettra de faire un traitement quelconque (par exemple : afficher l'image en grand).

Première étape, ajouter un **TLayout** et lui donner un **StyleName (ListItem)** et A.F.R.O. Puis, ajouter les différents éléments : un **TLayout** cadré à gauche contenant un **TImage** ('Icon') centré, un **TLayout** cadré à droite contenant un bouton ('Info') et enfin un dernier **TLayout** qui prendra le reste de la place (**alClient**) et contiendra deux **TLabels** ('Size' et 'Text').
L'exercice est assez délicat, la marche à suivre est cependant la même que pour notre style de début. N'oubliez pas la manipulation **A.F.R.O** (il m'est arrivé de ne plus voir du tout le dessin de mon style et d'être obligé de recommencer malgré tout).



Une fois le style terminé, passons à la codification.

```
procedure TForm3.Button3Click(Sender: TObject);
var Info   : TSearchRec;  // structure de
recherche
    chemin : String;      // répertoire
sélectionné
    Item: TListBoxItem;   // item de la liste

begin
// Note : SelectFolder est une fonction se
trouvant dans une unité
// séparée : winfolderselectutils.pas
 MonLabelStylé.Text := SelectFolder('Sélectionnez
le répertoire');

 Chemin :=
IncludeTrailingPathDelimiter(label1.Text);
 try
   Listbox1.BeginUpdate;   // passage en mode
update
  // efface une précédente recherche si
nécessaire
   Listbox1.Items.Clear;
  { Recherche de la première entrée du répertoire
}
  If FindFirst(Chemin+'*.jpg',faAnyFile,Info)=0
Then
  Begin
    Repeat
       If (Info.Attr And faDirectory) = 0
        then
           begin
            // Création de l'item
            Item := TlistBoxItem.Create(nil);
            // affectation à la liste
            Item.Parent := ListBox1;
            // Stockage nom complet
            Item.TagString := Chemin+Info.Name;
            Item.StyleLookup := 'ListeItem';   //
style de l'item
            Item.Text := info.name;            //
nom du fichier
           // Hauteur de l'item (pour ne pas
prendre la hauteur par défaut)
            Item.Height := 60;
          try
```

```
            // remplissage de l'image
(utilisation de vignette)

Item.ItemData.Bitmap.LoadThumbnailFromFile(Item.T
agString,Item.Height,Item.Height);
          except
          end;
            // Taille de l'image si > 1Mo
            if info.Size > SizeM
             // en Méga Octets
             then Item.StylesData['Size.Text'] :=
Format('%3.3f Mo',[info.Size/sizem])
             // sinon en Kilo Octets
             else Item.StylesData['Size.Text'] :=
Format('%3.3f Ko',[info.Size/sizek]);
            // très utile pour la sélection de
l'item
            Item.StylesData['info.tag'] :=
ListBox1.Items.Count-1;
            // affectation de l'événement au
bouton info de l'item
            Item.StylesData['info.OnClick'] :=
TValue.From<TNotifyEvent>(DoInfoClick); //
OnClick du bouton
          end;
     Until FindNext(Info)<>0;  // recherche
    FindClose(Info);           // fin de
recherche
  End;
  finally
    ListBox1.EndUpdate;        // fin de mise à
jour, force le dessin
  end;
end;
```

En plus des commentaires, quelques explications.
Concernant l'affectation des valeurs aux labels, pour le style précédent, j'avais utilisé une astuce en donnant comme **StyleName** la valeur 'Texte' (astuce que j'ai d'ailleurs réutilisée pour le nom du fichier) ; mais comment faire lorsque l'on a plusieurs **Labels** ? On récupère la propriété via **StylesData['nom de Style.propriété']**.
Pour l'image, je n'ai pas accédé à la propriété Bitmap de la même manière, mais par l'intermédiaire de **ItemData.Bitmap**.
Enfin, notez l'apparition d'une nouvelle propriété pour un item de liste : **TagString**, très pratique pour le stockage du nom complet de l'image.

5. Conclusion

Dans ce premier tutoriel, je n'ai fait qu'effleurer le sujet. Je n'ai pas abordé les styles déjà fournis dans des fichiers *.style et me suis cantonné à des styles contenus dans une ressource (en fait, inclus dans le dfm de la forme). Pour aller plus loin, je vous renvoie à une partie de ma liste de lectures/visionnages. Je vous conseille également d'installer le petit add-on de Marco Cantù (Lien 132) permettant de voir le source des styles. En effet, sous XE4, le composant **StyleBook** stocke les informations de style dans un format binaire compressé, rendant la chose plus difficile à éditer manuellement.
Source complet du programme : Lien 133

Retrouvez l'article de Serge Girard en ligne : Lien 134

Liens

Lien 01 : http://jeux.developpez.com/tutoriels/tile-mapping-construction-niveau/fichiers/tilesprogs.zip
Lien 02 : http://jeux.developpez.com/tutoriels/tile-mapping-construction-niveau/presentation-generale/
Lien 03 : http://monogame.codeplex.com/
Lien 04 : http://www.microsoft.com/visualstudio/fra/downloads#d-2013-express
Lien 05 : http://build.monogame.net/job/develop-win/lastSuccessfulBuild/artifact/Installers/Windows/
Lien 06 : http://tahe.developpez.com/dotnet/csharp/
Lien 07 : http://www.spriters-resource.com/
Lien 08 : http://franckh.developpez.com/tutoriels/csharp/monogame/part-I/images/Pacman-images.zip
Lien 09 : http://msdn.microsoft.com/fr-fr/library/microsoft.xna.framework.graphics.spritebatch.begin%28v=xnagamestudio.40%29.aspx
Lien 10 : http://franckh.developpez.com/tutoriels/csharp/monogame/part-I/fichiers/Pacman.zip
Lien 11 : http://franckh.developpez.com/tutoriels/csharp/monogame/part-I/
Lien 12 : http://djibril.developpez.com/tutoriels/linux/nagios-pour-debutant/
Lien 13 : http://perl.developpez.com/faq/perl/?page=sectionA32
Lien 14 : http://djibril.developpez.com/tutoriels/perl/installation-modules/
Lien 15 : http://www.developpez.net/forums/f476/autres-langages/perl/modules/
Lien 16 : http://ram-0000.developpez.com/tutoriels/reseau/Net-SNMP-1/
Lien 17 : http://ram-0000.developpez.com/tutoriels/reseau/Net-SNMP-2/
Lien 18 : http://djibril.developpez.com/tutoriels/linux/nagios-pour-debutant/#LIII-B-1
Lien 19 : http://wiki.monitoring-fr.org/nagios/addons/nsclient
Lien 20 : http://nsclient.org/nscp/
Lien 21 : http://djibril.developpez.com/tutoriels/linux/nagios-pour-debutant/
Lien 22 : http://djibril.developpez.com/tutoriels/linux/nagios-services/#LIV
Lien 23 : http://www.nsclient.org/nscp/wiki/CheckSystem/checkUpTime
Lien 24 : http://djibril.developpez.com/tutoriels/linux/nagios-services/
Lien 25 : http://gcompris.net/
Lien 26 : http://subsurface.hohndel.org/fr/
Lien 27 : http://www.wireshark.org/
Lien 28 : http://www.lxde.org/
Lien 29 : http://harmful.cat-v.org/software/c++/linus
Lien 30 : http://www.joelonsoftware.com/articles/fog0000000069.html
Lien 31 : http://www.google-melange.com/
Lien 32 : https://blog.wireshark.org/2013/10/switching-to-qt/
Lien 33 : http://blog.lxde.org/?p=1046
Lien 34 : http://sourceforge.net/p/gcompris/mailman/message/31951393/
Lien 35 : http://mirror.linux.org.au/linux.conf.au/2014/Thursday/83-Gtk_to_Qt_-_a_strange_journey_-_Dirk_Hohndel.mp4
Lien 36 : http://www.developpez.net/forums/d1417107/c-cpp/bibliotheques/qt/vent-migration-nombreuses-applications-gtk-passent-qt/
Lien 37 : http://blog.qt.digia.com/?p=37640
Lien 38 : http://qt-project.org/doc/qt-5/qtwebkit-index.html
Lien 39 : http://qt-project.org/doc/qt-5/qtbluetooth-index.html
Lien 40 : http://qt-project.org/doc/qt-5/qtnfc-index.html
Lien 41 : http://qt-project.org/doc/qt-5/qtquick-index.html
Lien 42 : http://qt-project.org/doc/qt-5/qtsensors-index.html
Lien 43 : http://qt-project.org/doc/qt-5/qtmultimedia-index.html
Lien 44 : http://qt-project.org/doc/qt-5/qtandroidextras-index.html
Lien 45 : http://qt-project.org/doc/qt-5/qtquickcontrols-index.html
Lien 46 : http://qt-project.org/doc/qt-5/qkeysequenceedit.html
Lien 47 : http://qt-project.org/doc/qt-5/accessible.html
Lien 48 : http://qt-project.org/doc/qt-5/qtwinextras-index.html
Lien 49 : http://qt-project.org/doc/qt-5/qtmacextras-index.html
Lien 50 : http://qt-project.org/doc/qt-5/qtimezone.html
Lien 51 : http://qt-project.org/doc/qt-5/qcollator.html
Lien 52 : http://qt.developpez.com/actu/60759/Qt-5-2-nouvelle-implementation-du-graphe-de-scene-Qt-Quick-avec-de-tres-grandes-ameliorations-de-performances/
Lien 53 : http://qt-project.org/doc/qt-5/qtquick-statesanimations-topic.html#animators
Lien 54 : http://blog.qt.digia.com/blog/2013/12/12/qt-creator-3-0-released/
Lien 55 : http://qt-project.org/doc/qt-5/qtpositioning-index.html
Lien 56 : http://qt-project.org/doc/qt-5/qtbluetooth-index.html
Lien 57 : http://qt-project.org/doc/qt-5/qtnfc-index.html
Lien 58 : http://qt-project.org/doc/qt-5/qtwinextras-index.html
Lien 59 : http://qt-project.org/doc/qt-5/qtandroidextras-index.html
Lien 60 : http://qt-project.org/doc/qt-5/qtwebkit-index.html
Lien 61 : https://play.google.com/store/apps/details?id=org.qtproject.quickforecast&hl=en
Lien 62 : https://itunes.apple.com/us/app/quick-forecast/id736658981?ls=1&mt=8
Lien 63 : http://qt-project.org/downloads
Lien 64 : http://www.developpez.net/forums/d1382077/c-cpp/bibliotheques/qt/sortie-qt-5-2-1-a/#post7671258
Lien 65 : http://qt.developpez.com/actu/62690/Qt-5-2-facilitera-fortement-le-deploiement-vers-les-plateformes-Android-avec-une-automatisation-de-la-creation-des-paquets-dans-Qt-Creator-3-0/
Lien 66 : http://download.qt-project.org/official_releases/qtcreator/3.0/3.0.0/
Lien 67 : http://qt.digia.com/qt52
Lien 68 : http://www.developpez.net/forums/d1388157/c-cpp/bibliotheques/qt/edi/qt-creator/sortie-qt-creator-3-0-1-a/#post7610459
Lien 69 : http://kdab.developpez.com/tutoriels/opengl-qt-5.1/01-fonctions-extensions/
Lien 70 : http://qt-project.org/doc/qt-5.0/qtgui/qopenglbuffer.html
Lien 71 : http://qt-project.org/doc/qt-4.8/qglbuffer.html
Lien 72 : http://www.opengl.org/wiki/Vertex_Specification#Vertex_Array_Object
Lien 73 : http://qt-project.org/doc/qt-5.1/qtgui/qopenglvertexarrayobject.html
Lien 74 : http://doc-snapshot.qt-project.org/qt5-dev/qtgui/qopenglvertexarrayobject.html

Lien 75 : http://kdab.developpez.com/tutoriels/opengl-qt-5.1/02-tableaux-coordonnees-vao/
Lien 76 : http://qt.developpez.com/actu/60759/Qt-5-2-nouvelle-implementation-du-graphe-de-scene-Qt-Quick-avec-de-tres-grandes-ameliorations-de-performances/
Lien 77 : http://qt.developpez.com/actu/63087/Qt-5-2-Beta-propose-un-support-complet-d-Android-et-iOS-ainsi-que-des-modules-pour-l-acces-a-des-fonctionnalites-natives/
Lien 78 : http://qt.developpez.com/actu/64973/Sortie-de-Qt-5-2-RC1-portant-a-un-excellent-niveau-de-qualite-les-nouvelles-fonctionnalites-support-d-Android-iOS-NFC-Bluetooth-u2026/
Lien 79 : http://www.developpez.net/forums/d1406269/autres-langages/python-zope/gui/pyside-pyqt/sortie-pyqt-5-binding-riverbank-poursuit-evolution/
Lien 80 : https://blogs.oracle.com/java-platform-group/entry/java_8_will_use_tls
Lien 81 : http://docs.oracle.com/javase/tutorial/essential/concurrency/forkjoin.html
Lien 82 : http://www.developpez.net/forums/d1208364-2/java/general-java/oracle-devoile-roadmap-jdk-8-a/#post7675694
Lien 83 : http://www.jetbrains.com/
Lien 84 : http://www.jetbrains.com/idea/
Lien 85 : http://www.eclipse.org/
Lien 86 : http://www.netbeans.org/
Lien 87 : http://www.jetbrains.com/idea/features/editions_comparison_matrix.html
Lien 88 : http://www.jetbrains.com/idea/download/
Lien 89 : http://www.eclipse.org/mylyn/
Lien 90 : http://www.jetbrains.com/idea/features/tasks_and_context.html
Lien 91 : http://www.jetbrains.com/idea/features/index.html
Lien 92 : https://developers.google.com/events/io/
Lien 93 : http://www.gradle.org/
Lien 94 : https://code.google.com/p/spock/
Lien 95 : http://www.scala-sbt.org/
Lien 96 : http://nodejs.org/
Lien 97 : https://github.com/karma-runner/karma
Lien 98 : https://www.dartlang.org/
Lien 99 : http://www.w3.org/TR/components-intro/
Lien 100 : http://mustache.github.io/
Lien 101 : http://handlebarsjs.com/
Lien 102 : http://www.jetbrains.com/idea/whatsnew/index.html
Lien 103 : http://jetbrains.dzone.com/articles/top-20-code-completions-in-intellij-idea
Lien 104 : http://www.jetbrains.com/idea/features/code_completion.html
Lien 105 : http://www.jetbrains.com/idea/features/refactoring.html
Lien 106 : https://github.com/joel-costigliola/assertj-core
Lien 107 : http://www.jetbrains.com/idea/documentation/ssr.html
Lien 108 : http://www.jetbrains.com/idea/docs/IntelliJIDEA_ReferenceCard.pdf
Lien 109 : http://www.jetbrains.com/idea/docs/IntelliJIDEA_ReferenceCard_Mac.pdf
Lien 110 : http://www.ptxstore.com/jetbrains/product_info.php?products_id=1638
Lien 111 : http://plugins.jetbrains.com/plugin/6546
Lien 112 : http://www.jetbrains.com/idea/features/eclipse_java.html
Lien 113 : http://java.developpez.com/faq/intellijidea
Lien 114 : http://www.developpez.net/forums/f1871/java/edi-outils-java/autres-edi/intellij/
Lien 115 : http://confluence.jetbrains.com/display/IntelliJIDEA/IntelliJ+IDEA+for+Eclipse+Users
Lien 116 : http://confluence.jetbrains.com/display/IntelliJIDEA/IntelliJ+IDEA+for+NetBeans+Users
Lien 117 : http://linsolas.developpez.com/articles/java/outils/intellij-idea-13/
Lien 118 : http://eclipse.org/org/press-release/20140203_foundation_anniversary.php
Lien 119 : http://www.developpez.net/forums/d1413937/club-professionnels-informatique/actualites/fondation-eclipse-celebre-dixieme-anniversaire/
Lien 120 : http://dmouronval.developpez.com/tutoriels/css/cascade-heritage-specificite/#LI-A
Lien 121 : http://dmouronval.developpez.com/tutoriels/css/cascade-heritage-specificite/
Lien 122 : http://www.synbioz.com/blog/mise_en_page_flexbox
Lien 123 : http://caniuse.com/#search=flex
Lien 124 : http://synbioz.developpez.com/tutoriels/css/mise-en-page-flexbox/
Lien 125 : http://www.js-attitude.fr/js-puissant/
Lien 126 : http://www.git-attitude.fr/
Lien 127 : http://www.html5rocks.com/en/tutorials/casestudies/technitone/
Lien 128 : http://www.git-attitude.fr/2010/07/18/heberger-un-serveur-git-avec-gitosis-linux-osx/
Lien 129 : http://blog.stevenlevithan.com/archives/javascript-regex-lookbehind
Lien 130 : http://javascript.developpez.com/tutoriels/maitriser-expressions-rationnelles/
Lien 131 : http://www.tindex.net/FireMonkey/Styles.html
Lien 132 : http://cc.embarcadero.com/item/29428
Lien 133 : http://delphi.developpez.com/telecharger/detail/id/3872/Premier-pas-avec-les-Styles-Firemonkey-avec-XE4
Lien 134 : http://delphi.developpez.com/tutoriels/firemonkey/intro-styles-firemonkey-xe4/