



Developpez

Le Mag

Édition de Juin - Juillet 2011.

Numéro 34.

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

Java	Page 2
Android	Page 6
PHP	Page 13
Développement Web	Page 22
Web sémantique	Page 26
C/C++/Gtk+	Page 35
Qt	Page 44
Visual Basic	Page 52
Liens	Page 59

Article Web sémantique



Le tutoriel SPARQL

Ce tutoriel nous donne un cours rapide sur SPARQL. Il couvre toutes les fonctionnalités majeures du langage de requête à travers des exemples.

traduit par **Thibaut Cuvelier et Julien Plu**

Page 26



Article Android

Éditorial

Ce mois-ci, la rubrique Web sémantique fait une entrée fracassante dans le magazine, pour le plus grand plaisir de tous.

Profitez-en bien !

La rédaction

Introduction à la programmation sous Android

Ce tutoriel a pour but de vous présenter succinctement Android, ainsi que les prémices de la programmation sous celui-ci.

par **Nazim Benbourahla**

Page 6



Le Java Community Process approuve à contrecœur Java SE 7

Des voix s'élèvent contre la politique de licence imposée par Oracle

Le scrutin sur Java SE 7 par le Java Community Process vient d'avoir lieu et d'approuver à la majorité écrasante cette prochaine édition standard du langage, mais non sans de nombreux commentaires fustigeant notamment la politique de licence menée par Oracle.

Google, bien que satisfait de l'avancement de la technologie sur cette version, mais visiblement décidé à gâcher la fête d'Oracle, a manifesté le seul vote négatif, contre 13 voix positives et 2 abstentions.

IBM, Red Hat, SouJava, London Java Community, Goldman Sachs et Fujitsu ont tous voté oui tout en affirmant dans les commentaires du vote leur refus de la politique de licence de Java, leur scepticisme envers le groupe d'experts et la transparence de tout le processus.

Sans mentionner le désaccord ayant conduit la fondation Apache à quitter le Process ([Lien 01](#)) ni le procès en cours ([Lien 02](#)), Google rappelle que le kit de validation TCK (Technology Compatibility Kit), partie intégrante du standard, ne doit pas être utilisé pour interdire les implémentations compatibles, notamment dans le domaine du mobile.

L'éditeur d'Android, numéro 1 mondial des OS mobiles, juge que « *les licences qui contiennent de telles restrictions sont incompatibles avec les exigences du JSPA, et violent les attentes de la communauté Java (sic) que les spécifications du JCP puissent être implémentées ouvertement* ».

Werner Keil, expert en Java et membre à part entière du JCP, justifie son abstention par le « *manque de transparence* » tant dans la gestion de l'ensemble Umbrella JSR (Java Specification Request) que pour d'autres composants essentiels, en particulier le projet Coin.

Pour rappel, Java SE 7 offrira notamment une meilleure compatibilité avec les processeurs multicœurs, des possibilités de scriptage dynamique, une API unifiée pour accéder aux systèmes du fichier de l'OS tout en continuant d'autoriser les opérations spécifiques à chaque plateforme.

Oracle, qui a naturellement voté oui sans commentaire, se heurte à de vives critiques qui ne se sont pas tassées malgré la volonté que le géant a récemment manifestée de révolutionner le fonctionnement du Java Community Process vers plus de transparence, de réactivité et d'agilité ([Lien 03](#)).

Source : [Lien 04](#)

Commentez cette news d'Idelways en ligne : [Lien 05](#)

Java : la machine virtuelle JRockit d'Oracle devient gratuite

Pour le développement et l'usage interne en production

Oracle, propriétaire d'une pléthore de machines virtuelles à la suite de son rachat de Sun, travaille depuis l'année passée à la fusion de JRockit et Hotspot en une seule machine virtuelle dans le cadre du projet OpenJDK ([Lien 06](#)).

En ce sens, la machine virtuelle JRockit devient désormais gratuite pour « *le développement et l'utilisation interne en production* » d'après un billet de blog par Henrik Ståhl, chef de produit JRockit à Oracle.

Tout en restant propriétaire et son code source non accessible, Ståhl fait savoir que des développeurs d'Oracle portent progressivement des idées et fonctionnalités de JRockit à la version libre de Java OpenJDK.

JRockit est disponible gratuitement sous une nouvelle licence, dérivée de la « *Sun Binary Code Licence* ».

Les fonctionnalités de hautes performances et disponibilité, y compris JRockit Mission Control, JRockit Real Time et JRockit Virtual Edition, nécessitent toujours une licence commerciale.

Ces composants, préalablement disponibles uniquement en compagnie des produits d'Oracle (comme le serveur WebLogic), peuvent être achetés indépendamment et utilisés désormais pour n'importe quel projet Java.

Pour mémoire, JRockit a été initialement développée par Appeal Virtual Machines, rachetée ensuite par BEA Systems avant de faire partie d'Oracle Fusion Middleware depuis 2008.

JRockit est disponible en téléchargement pour Windows, Solaris et Linux sur cette page : [Lien 07](#).

Source : blog de Henrik Ståhl ([Lien 08](#))

Commentez cette news d'Idelways en ligne : [Lien 09](#)

Sortie de Scala 2.9.0

Cette version permettra-t-elle au langage de se propulser dans le monde de l'entreprise ?

L'équipe de développement de Scala a annoncé la sortie de la version 2.9.0 du langage.

Parmi les nouveautés et améliorations nous trouvons :

- **les collections parallèles** : chaque collection a un pendant parallèle permettant d'effectuer les opérations telles que *map* ou *filter* en parallèle ;
- le trait **App** permettant de remplacer l'ancien trait

(déprécié) **Application** qui n'était pas thread safe ;

- de nouvelles façons d'exécuter des programmes Scala avec le lanceur :
 - `scala <jarfile>` similaire à `java -jar` ;
 - `scala <classname>` exécute la méthode `main` de l'objet ;
 - `scala <sourcefile>` exécute le script. Si le contenu n'est pas un script, trouve un objet avec une méthode `main` et l'exécute ;
 - `scala -save <sourcefile>` crée un jar avec les sources compilées. Ce jar peut ensuite être exécuté via `scala <jarfile>` ;
- améliorations des performances.

Plus d'informations : [Lien 10](#).

À noter aussi le plugin Eclipse qui suit plus ou moins le

cycle de releases du langage, qui a refondu son implémentation permettant de meilleures performances et une meilleure stabilité. Plus d'informations : [Lien 11](#).

Dernière nouvelle liée à la sortie de cette version 2.9.0 de Scala : Martin Odersky, créateur du langage, a fondé avec des membres actifs de la communauté la société Typesafe ([Lien 12](#)) dans le but de fournir une suite complète open source de développement Scala qui inclut pour le moment Scala 2.9.0 et Akka 1.1 ([Lien 13](#)) et devrait inclure la nouvelle version de scala-ide ([Lien 14](#)). La société fournit aussi des formations et du support professionnels aux entreprises.

Cette nouvelle version devrait permettre au langage de se propulser dans le monde de l'entreprise et de passer du côté des langages mainstream.

Commentez cette news de George7 en ligne : [Lien 15](#)

Les derniers tutoriels et articles

Les bundles en GWT

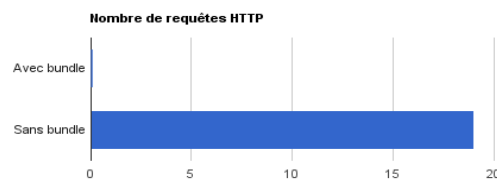
Cet article va présenter le concept de client bundle GWT et montrer son utilisation. Prérequis : utilisation de GWT.

1. Qu'est-ce qu'un bundle de données ?

Une application web est accompagnée d'un nombre plus ou moins important de ressources statiques : notamment les images et feuilles de style CSS. Chaque référence à l'une de ces ressources va provoquer son chargement via une requête HTTP dédiée. On peut facilement arriver à plus de cinquante images (icônes ou éléments de style graphique) pour une seule page. Dans les styles modernes il n'est pas rare de manipuler des 9-box : un conteneur graphique utilisant neuf images (les quatre coins, les quatre bords et le centre). Chaque type de 9-box nécessitera neuf requêtes HTTP.

L'optimisation d'un site web réside en partie dans sa vitesse d'affichage et la minimisation des requêtes HTTP est un axe de travail. Si on avait une seule grosse image concaténant toutes les petites images utilisées par notre thème graphique, il ne faudrait déjà plus qu'une seule requête HTTP pour récupérer tous les éléments de style. Et on peut faire le même raisonnement pour minimiser les inclusions de feuilles de style en les concaténant. C'est cette facilité de regroupement qui est amenée par les bundles de données GWT.

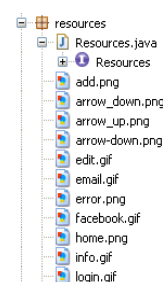
Voici une comparaison du nombre de requêtes HTTP du temps de chargement et pour une page utilisant un bundle de données et la même page ne l'utilisant pas. Bien sûr le temps de chargement dépend de la bande passante, de la disponibilité du serveur, de la vitesse du poste client... mais les graphiques suivants montrent l'ordre de grandeur.



Les temps de chargement sont légèrement diminués dans le cas d'un bundle et le nombre de requêtes HTTP d'images dans cet exemple est passé de 19 à 0 ! En effet sur mon navigateur de test, GWT a compilé les images en base64 et elles sont directement incluses dans le code JavaScript généré. Notons au passage qu'il aurait quand même fallu une requête pour charger le bundle sur les navigateurs ne supportant pas la définition des images en base64.

2. Images

Le regroupement d'images est le plus intuitif. GWT va agencer dans une nouvelle image PNG toutes les images qui lui sont données à gérer. Il y a cependant une spécificité par rapport aux fichiers JPG puisque ceux-ci seront mis dans des fichiers séparés pour minimiser la taille de l'image générée. Dans votre projet GWT il vous suffit de déclarer toutes les images à utiliser dans une interface étendant `ClientBundle`. Créez un nouveau package contenant cette interface et toutes vos images.



Inscrivez chaque image à gérer dans l'interface

```
ImageResource icon();
```

Ajoutez le constructeur GWT du bundle :

```
public static ImageBundle R =
GWT.create(ImageBundle.class);

public interface Resources extends ClientBundle{
    public static Resources R =
GWT.create(Resources.class);

    ImageResource add();
    ImageResource edit();
    ImageResource email();
    ImageResource facebook();
    ImageResource login();
    ImageResource logout();
    ImageResource refresh();
    ImageResource remove();
    ImageResource rss();
    ImageResource twitter();
    ImageResource user_group();
    ...
}
```

La création d'une signature de fonction "icon" correspondra à une image icon.bmp, icon.png, icon.jpg ou icon.gif située dans le même package. Le plugin GWT de votre environnement vous avertira si vous référencez une image inexistante dans le package. Si vous souhaitez que la fonction "icon" soit en réalité associée à une autre image, il va falloir le préciser avec une annotation @Source("monImage.png"). Voilà le résultat de la compilation des images :



Les images contenues dans le bundle s'utiliseront en indiquant la fonction de l'interface qui leur est associée. Voilà par exemple comment instancier des objets GWT image liés au bundle. Et bien sûr au final, notre page se charge très rapidement côté client, nous avons au plus une seule requête HTTP pour charger toutes les images.

```
RootPanel.get().add(new
Image(Resources.R.add()));
```

3. Feuilles de style

GWT amène une gestion des feuilles de style très puissante : il amène des notions de variables CSS, de factorisation de règles CSS, d'inclusions conditionnelles et bien sûr il permet d'utiliser les ressources images définies en bundle.

De la même manière que pour les bundles d'images, il vous suffit d'inclure votre fichier CSS dans un package de votre projet et de déclarer une méthode du même nom dans l'interface du bundle. Une méthode "style" sera associée au fichier "style.css". Exemple

```
public interface Resources extends ClientBundle{
    public static Resources R =
GWT.create(Resources.class);

    CssResource style();
}
```

La déclaration d'une méthode style() est associée au fichier style.css situé dans le package courant. L'inclusion de cette feuille de style se fait par

```
Resources.R.style().ensureInjected();
```

Les feuilles de style sont elles aussi optimisées, les classes sont regroupées, les commentaires sont supprimés. Les éléments de style peuvent même être directement inscrits dans un attribut style des éléments du DOM s'ils sont peu utilisés. L'utilisation classique peut s'illustrer avec cet exemple :

```
@external .test;
.test{
    font-size: 9pt;
    color: #985;
}
```

```
monWidget.addStyleName("test");
```

Pour utiliser directement les classes CSS dans votre code HTML, il faut déclarer ces classes CSS avec l'annotation @external. De cette manière les sélecteurs ne seront pas optimisés et ils resteront cohérents avec votre code HTML.

3.1. Utilisation des bundles d'images dans un bundle CSS

GWT permet de réutiliser dans une feuille de style une image incluse dans un bundle d'images. C'est là tout l'intérêt des sprites GWT. Rien ne vous empêche de créer un même bundle pour stocker à la fois des images et des feuilles de style. La déclaration d'une règle CSS utilisant un élément d'un bundle d'images se fait de la manière suivante :

```
@external .info;
@sprite
.info{
    gwt-image : "info";
}}
```

```
public interface Resources extends ClientBundle{
    public static Resources R =
GWT.create(Resources.class);

    CssResource style();
    ImageResource info();
}}
```

4. Utilisation des bundles de données texte

Bien qu'ils soient moins utiles, les bundles de données peuvent vous permettre d'optimiser encore un peu les transferts de données statiques de votre application. Si vous avez besoin de manipuler un fichier texte statique (fichier txt, xml, json...), les bundles de données texte sont pour vous.

La définition d'un bundle de données est similaire à celle des bundles CSS et images : vous ajoutez une méthode dans l'interface de votre bundle et GWT associera un fichier du package courant avec le nom de la méthode. Une méthode "methode" cherchera un fichier "methode.txt", à moins que vous n'ajoutiez l'annotation @source, comme pour les bundles CSS et images.

Il y a cependant une nuance pour les bundles de texte : ceux-ci peuvent être accédés de manière synchrone ou asynchrone. La méthode synchrone chargera le contenu du fichier texte dès le chargement de la page alors que la ressource asynchrone ne sera accédée et transférée que lors de son utilisation.

```
public interface Resources extends ClientBundle {
    public static Resources R =
    GWT.create(Resources.class);

    ImageResource dvp();
    CssResource style();

    TextResource countries(); // une ressource
    synchrone -> countries.txt
    ExternalTextResource countriesAsync(); // une
    ressource asynchrone -> countriesAsync.txt
}
```

```
// synchrone
String countries =
Resources.R.countries().getText();

// asynchrone
Resources.R.countriesAsync().getText(new
ResourceCallback<TextResource>(){
    @Override
    public void onError(ResourceException e)
```

```
{
    // gestion de l'erreur de
    récupération
}

@Override
public void onSuccess(TextResource
resource) {
    String countries =
resource.getText();
    // traitement des données,
    parsing ...
}
});
```

La récupération du contenu d'un bundle texte renvoie un objet String, à vous ensuite de le parser s'il s'agit d'un fichier XML, JSON ou autre.

5. Risques de dérive

La notion de bundle peut paraître très tentante, mais il faut bien faire attention à ne pas en abuser, vous pourriez perdre en performance. Le bundle idéal est celui dont toutes les ressources sont utilisées sur une page web. Il est donc déconseillé de stocker dans un même bundle des éléments graphiques utilisés sur des pages différentes ou visibles par des utilisateurs différents.

Une solution pertinente sera de créer un bundle centralisant toutes les ressources communes à tous les modules de votre application web et ensuite autant de bundles que de modules, pour y stocker les éléments spécifiques à chaque module. De cette manière vous maximisez vos chances d'utiliser tous les éléments d'un bundle quand il aura été téléchargé par le navigateur.

Retrouvez l'article de Pierre Schwartz en ligne : [Lien 16](#)

Introduction à la programmation sous Android

Ce tutoriel a pour but de vous présenter succinctement Android, ainsi que les prémices de la programmation sous celui-ci.

1. Android, c'est quoi ?

Android est un OS mobile Open Source pour smartphone, PDA, MP3 et tablette. Conçu initialement par Android Inc, il a été racheté par Google en 2005.

Pour commencer la programmation Android, il faut d'abord installer le **SDK Android** et comprendre les bases de la programmation sous Android. Puis nous allons faire notre premier programme sous Android c'est-à-dire le bien connu « **Hello Word** » pour bien comprendre ces bases.

2. Composantes d'une application Android

Une application Android est composée d'éléments de base :

2.1. Activities (Activités en français)

Une activité est la composante principale pour une application Android. Elle représente l'implémentation métier dans une application Android.

Prenant l'exemple d'une application qui liste tous vos fichiers mp3 présents dans votre téléphone, le projet pourrait se décomposer comme ci-dessous :

- une vue pour afficher la liste des mp3 ;
- une activité pour gérer le remplissage et l'affichage de la liste ;
- si l'on veut pouvoir rajouter, supprimer des mp3, on pourrait rajouter d'autres activités.

2.2. Services

Un service, à la différence d'une activité, ne possède pas de vue mais permet l'exécution d'un algorithme sur un temps indéfini. Il ne s'arrêtera que lorsque la tâche est finie ou que son exécution est arrêtée.

Il peut être lancé à différents moments :

- au démarrage du téléphone ;
- au moment d'un événement (arrivée d'un appel, SMS, mail, etc.) ;
- lancement de votre application ;
- action particulière dans votre application.

2.3. Broadcast and Intent Receivers

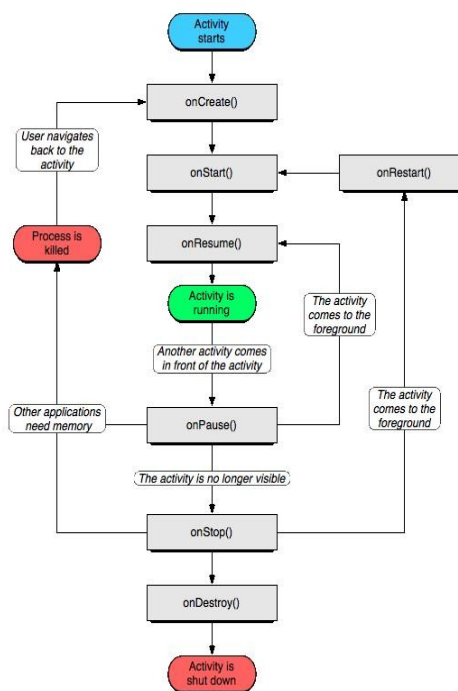
Un Broadcast Receiver comme son nom l'indique permet d'écouter ce qui se passe sur le système ou sur votre application et déclencher une action que vous aurez prédéfinie. C'est souvent par ce mécanisme que les services sont lancés.

2.4. Content providers

Les « content providers » servent à accéder à des données depuis votre application. Vous pouvez accéder :

- aux contacts stockés dans le téléphone ;
- à l'agenda ;
- aux photos ;
- ainsi qu'à d'autres données depuis votre application grâce aux content providers.

3. Cycle de vie d'une application Android



3.1. onCreate

Cette méthode est appelée à la création de votre activité (Activity). Elle sert à initialiser votre activité ainsi que toutes les données nécessaires à cette dernière.

Quand la méthode **onCreate** est appelée, on lui passe un Bundle en argument. Ce Bundle contient l'état de sauvegarde enregistré lors de la dernière exécution de votre activité.

3.2. onStart

Cette méthode est appelée dans le cas où votre application est en arrière-plan et qu'elle repasse en avant-plan.

Si votre activité ne peut pas aller en avant-plan quelle que soit la raison, l'activité sera transférée à **onStop**.

3.3. onResume

Cette méthode est appelée après **OnStart** (au moment où votre application repasse en avant-plan).

OnResume est aussi appelée quand votre application passe en arrière-plan à cause d'une autre application.

3.4. onPause

Appelée juste avant qu'une autre activité que la vôtre passe en **OnResume**. À ce stade, votre activité n'a plus accès à l'écran, vous devez arrêter de faire toute action en rapport avec l'interaction utilisateur. Vous pouvez par contre continuer à exécuter des algorithmes nécessaires mais qui ne consomment pas trop de CPU.

3.5. onStop

Appelée quand votre activité n'est plus visible quelle que soit la raison.

3.6. onDestroy

Appelée quand votre application est totalement fermée (Processus terminé).

4. Installer votre environnement de développement

4.1. Installation du SDK Android

1. Pour commencer allez sur le lien suivant : [Lien 17](#) et téléchargez la version du SDK qui convient à votre OS. Pour la suite on est sur un Windows 7.
2. Vous avez le choix entre la version Zip ou la version exe. Dans cet exemple on a pris la version Zip.
3. Une fois le SDK téléchargé, allez dans le dossier où se trouve le fichier Zip et l'extraire dans le dossier de votre choix.
4. Lancez l'exécutable « SDK Setup » qui se trouve à la racine du dossier.
5. La fenêtre suivante apparaît (la dernière version actuelle du SDK est 2.3) :



6. Choisissez ce que vous voulez installer. Par exemple :
 - **SDK Platform Android 2.x** : correspond tout simplement au SDK Android basique en version 2.x ;
 - **Samples for SDK API 7** : correspond à quelques exemples ;
 - **Android + Google APIs** : correspond au SDK Android (1re option) + Google Api qui inclut différentes fonctions comme GoogleMap, etc. ;
 - **Galaxy Tab** : c'est le SDK pour la tablette Samsung Galaxy Tab.
 - En cas de problème d'installation, allez dans « **Settings** » et cochez « **Force https:// sources to**

be fetched using http:// ».

7. Cliquez sur « Install ».
8. Installez Eclipse : [Lien 18](#) (téléchargez une version comprenant Java, soit la version pour les développeurs Java ou la version pour les développeurs J2EE) : version installée pour ce tutoriel : 3.6.1 Helios.
9. Il faut aussi installer JDK (Java Development Kit) et JRE (Java Runtime Environment) si ce n'est pas déjà fait : [Lien 19](#).
10. Lancez votre Eclipse, allez dans le menu « Help and Install New Software »
11. Dans la partie « Available Software », cliquez sur « Add ».
12. Rajoutez le nom du site (« ADT plugin » par exemple). Dans la location rajoutez : [Lien 20](#), puis cliquez sur OK.
13. Revenez dans « Available Software », vous devez voir « Developer Tools ». Sélectionnez-le et appuyez sur « Next ».
14. Cliquez sur « Next » pour lire et accepter la licence et puis cliquez sur « Finish ».
15. Pour finir l'installation relancez Eclipse.

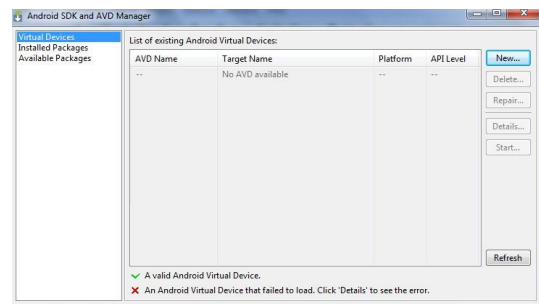
Voilà votre environnement de développement est prêt à être utilisé.

4.2. Configuration de votre environnement de développement

Vous avez dû remarquer qu'un nouvel élément est apparu dans votre Eclipse dans le menu du haut (un petit Android qui sort d'une boîte) :



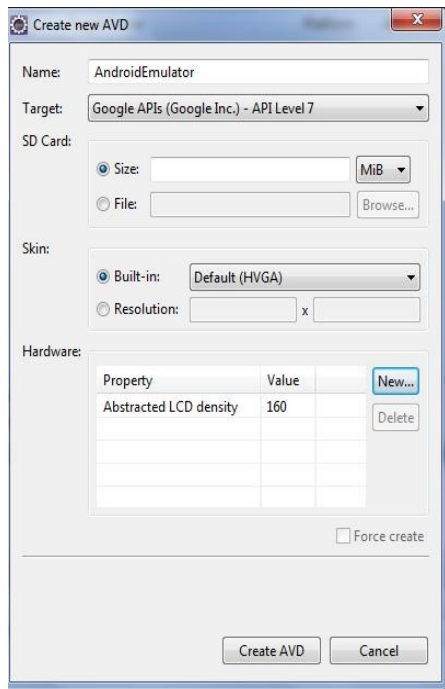
Cliquez sur l'icône en question, une nouvelle fenêtre va apparaître.



Cet écran vous permettra de :

- installer de nouveaux paquets (Available Packages) ;
- mettre à jour vos paquets ;
- voir les paquets déjà installés (Installed Packages) ;
- créer votre émulateur Android et cela grâce à l'onglet « Virtual Devices ». Cliquez sur le bouton « New ».

Une nouvelle fenêtre pour la création de votre émulateur apparaîtra.



- **Name** : le nom de votre émulateur (sans espace).
- **Target** : version du SDK Android de l'émulateur.
- **SD Card (facultatif)** : configuration de la SD Card (Taille, etc.).
- **Skins** : choisissez la taille, résolution de votre émulateur. Des émulateurs préconfigurés se trouvent dans la partie Built-in.
- **Hardware** : cette partie permet de rajouter le matériel spécifique à votre émulateur. Par exemple vous pouvez rajouter un GPS, configurer le clavier, l'accéléromètre, etc.

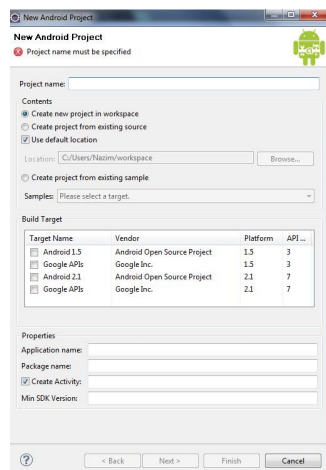
L'émulateur apparaîtra maintenant dans la liste des émulateurs disponibles.

Maintenant on va passer à la partie la plus intéressante.

5. Ma première application sous Android

5.1. Création du projet « Hello World »

Dans Eclipse, cliquez sur « File -> New -> Android Project ». La fenêtre ci-dessous s'affichera :



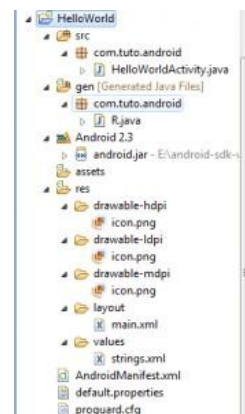
Remplissez les champs :

- **Project name** : le nom du projet. Pour notre exemple on choisira **Hello World**.
- **Build Target** : cochez la SDK que vous souhaitez. On prendra **Android 2.3**.
- **Properties** :
- -----**Application name** : le nom de l'application. On choisira Hello World ;
- -----**Package name** : le nom du package principal de l'application. Il faut que ce dernier comporte au moins deux identifiants séparés par des points. On prendra com.tuto.android ;
- -----**Create Activity** : si vous laissez coché, vous devez spécifier un nom pour l'activité de base de votre application. Nous choisirons HelloWorldActivity ;
- -----**Min SDK Version** : vous pouvez spécifier quelle version minimum du SDK est nécessaire pour le fonctionnement de votre application. Ce champ est facultatif.

Puis cliquez sur « Finish », le projet « Hello World » va apparaître dans l'arborescence d'Eclipse.

5.2. Explication de l'arborescence du projet

Voici le résultat de la création de votre projet et l'arborescence de ce dernier



- **src** : ce dossier contient les sources de votre application (code JAVA) et les packages.
- **com.tuto.android** : un package de votre application. Bien sûr, vous pouvez avoir plusieurs packages dans votre application.
- **HelloWorldActivity.java** : notre principale activité. (je vous conseille d'avoir plusieurs activités pour les différentes parties de votre code).
- **gen** : dossier qui contiendra le fichier R.java (ce fichier est généré automatiquement à partir de vos vues et fichiers de ressource).
- **R.java** : ce fichier est automatiquement généré par le SDK Android à chaque précompilation.
- **assets** : contient des données qui pourront être utilisées dans votre application (images, vidéos, licence, etc.).
- **res** : c'est le dossier qui contiendra les ressources de votre application (images, vidéos, styles).
- **drawable-hdpi** : contient toutes les images, bitmaps dont vous avez besoin pour votre

- application en haute résolution.
- **drawable-ldpi** : contient toutes les images, bitmaps dont vous avez besoin pour votre application en basse résolution.
- **drawable-mdpi** : contient toutes les images, bitmaps dont vous avez besoin pour votre application en moyenne résolution.
- **Icon.png** : l'icône de votre application, cette icône sera affichée sur le bureau.
- **layout** : le SDK Android offre une technique de création d'interfaces graphiques à l'aide de fichiers XML. C'est dans ce dossier que vous incluez l'ensemble des fichiers décrivant vos interfaces. Vous pouvez créer d'autres dossiers pour les menus par exemple.
- **Main.xml** : le fichier principal de votre interface.
- **values** : ce dossier contient un ensemble de fichiers décrivant les valeurs (pseudovariables) utilisées par votre application. On peut, par exemple, y mettre des chaînes de caractères (strings.xml), des tableaux (arrays.xml), des entiers, des couleurs, etc.
- **Strings.xml** : fichier qui contient vos déclarations de chaînes de caractères.
- **AndroidManifest.xml** : définit le comportement de votre application au système Android. Ce fichier définit par exemple le nom, l'icône, la version min du SDK, les activités, les services, etc.

5.3. Hello, World!

Le projet exemple créé de base par Eclipse représente un « **Hello World!** ». Vous pouvez le lancer en tant qu'une application Android. Pour cela, il vous suffit de cliquer droit sur le projet, puis sélectionner l'option « Run As -> Android Application » et la l'émulateur devrait se lancer. L'émulateur prendra un peu de temps à se lancer la première fois (ne le fermez pas entre vos différentes modifications).

Voilà le deuxième écran qui devrait s'afficher si tout se passe bien (le premier est pareil mais avec juste écrit « Android »)



Notre Hello Word est bien fonctionnel mais reste encore à le comprendre. Allons voir le code pour comprendre ce qui se passe.

• AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
  xmlns:android="http://schemas.android.com/apk/res/android"
  package="com.tuto.android"
  android:versionCode="1"
  android:versionName="1.0">
  <application android:icon="@drawable/icon"
    android:label="@string/app_name">
    <activity android:name=".HelloWorldActivity"
      android:label="@string/app_name">
      <intent-filter>
        <action
          android:name="android.intent.action.MAIN" />
        <category
          android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
  </application>
</manifest>
```

- La balise « **manifest** » à la ligne deux contient plusieurs arguments, le plus important est « package », qui donne le nom du package dans lequel se trouve votre activité principale.
- La balise « **application** » sert à la déclaration de différentes propriétés de votre application :
 - -----**android:icon** : l'emplacement où se trouve l'icône de votre application ;
 - -----**android:label** : le nom de votre application (il se trouve dans strings.xml).
- La balise « **activity** » permet de déclarer une activité, à chaque nouvelle activité il faut remettre cette balise.
 - -----**android:name** : le nom de la classe Java qui représente l'activité. Le nom doit commencer par un . et on ne met pas le .java à la fin.
 - -----**android:label** : le label de l'activité en question.
 - -----**intent-filter** : c'est pour spécifier une action.
 - -----la sous-balise action est pour spécifier l'action à exécuter, dans notre cas c'est le main.
 - -----la sous-balise category est là pour spécifier la catégorie de l'action.
 - -----Voici un lien qui explique les différents types d'actions et de catégories : [Lien 21](#).
- **strings.xml**

```
<xml version="1.0" encoding="utf-8">
<resources>
<string name="hello">
Hello World, HelloWorldActivity!
</string>
<string name="app_name">Hello World</string>
</resources>
```

- dans les balises **resources**, on met une balise string à chaque fois que l'on a besoin de déclarer une chaîne de caractères ;
- on déclare deux chaînes :

- -----la chaîne **hello** qui contiendra « Hello World, HelloAndroidActivity! » qui est le message qui sera affiché dans l'application ;
- -----la chaîne **app_name** qui contient « Hello Andro » qui représente le nom de l'application.
- **main.xml**
- ----- On dispose de deux modes de visualisation.
- -----**Onglet Layout** : mode visualisation et édition d'interface ;
- -----**Onglet main.xml** : mode code source.
- -----On commence par une balise qui définit le layout : ici **LinearLayout**.
- -----Voici un lien pour la liste des différents layouts : [Lien 22](#).
- -----On déclare une composante **TextView** pour afficher du texte et on
- lui dit qu'elle doit afficher le contenu de **@string/hello**, donc de la
- variable **hello** qui est déclarée dans **strings.xml** c'est-à-dire « **Hello World, HelloWorldActivity!** ».
- **HelloAndroidActivity.java**

```
package com.mti.android;

import android.app.Activity;
import android.os.Bundle;

public class HelloAndroidActivity extends Activity
{
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

- Notre main activité. Elle doit hériter de la classe Activity ou d'une sous-classe de cette dernière.
- [Lien 23](#).
- La méthode « **OnCreate** » est équivalente au main, elle est appelée à la création de votre vue.
- On appelle simplement le **OnCreate** de la classe mère puis on initialise la vue. Puis, on met dedans **R.layout.main**, c'est-à-dire la vue déclarée dans le fichier **main.xml**.
- À chaque fois que vous voyez « **R** », c'est-à-dire

que l'on utilise du code qui a été généré par les différents fichiers xml de ressources.

- « **R.layout** » : on va chercher la vue déclarée dans le dossier layout et qui s'appelle **main** donc notre **main.xml**.
- R.java

```
/* AUTO-GENERATED FILE. DO NOT MODIFY.
 *
 * This class was automatically generated by the
 * aapt tool from the resource data it found. It
 * should not be modified by hand.
 */

package com.mti.android;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
}
```

Vous ne devez pas toucher à ce fichier, il est généré automatiquement à partir de vos fichiers qui se trouvent dans le dossier des ressources (res).

- Vous remarquez que toutes les variables déclarées dans strings.xml sont présentes, que l'interface déclarée dans main.xml aussi. Ce qui explique l'utilisation de la ligne R.layout.main dans le HelloWorldActivity.java ainsi que l'icône de l'application.

6. Conclusion

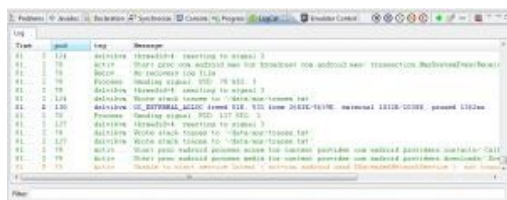
Voilà on s'arrête ici pour ce premier tutoriel, d'autres tutoriels vont suivre très prochainement et aborderont des sujets plus approfondis.

Retrouvez l'article de Nazim Benbourahla en ligne : [Lien 24](#)

Comprendre et corriger les erreurs dans votre application

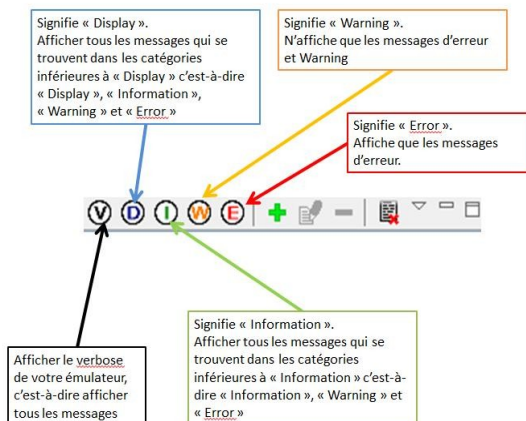
Ce tutoriel a pour but de vous aider à bien comprendre les messages d'erreurs que vous pouvez rencontrer lors du développement de votre application Android et les outils à votre disposition pour résoudre vos erreurs.

1. Logcat



Le **LogCat** trace l'exécution de l'émulateur et donc, de votre application pas à pas. Vous allez voir tout ce que fait l'émulateur et tout ce qu'il affiche.

Si vous ne disposez pas de l'onglet **LogCat**, il suffit d'aller dans le menu "Window -> Show View -> Other", puis choisissez LogCat.



En cas d'erreur dans votre application, elle sera affichée dans le LogCat et le fichier depuis lequel l'exception est lancée sera indiqué, ainsi que la nature de l'exception. Vous pouvez filtrer les logs via les différents niveaux par ordre croissant (debug, info, warning, error) quand vous êtes en train de déboguer votre application. Par exemple si vous choisissez de filtrer les logs à partir des warnings, vous n'aurez dans ceux-ci que les warnings et les errors. Lors de votre débogage et si vous voulez afficher des messages dans votre LogCat, il suffit d'utiliser :

```
Log.i ("TAG", "message");
```

La lettre qui vient après le Log dépend de l'importance de votre affichage, c'est-à-dire :

- i pour information, d pour display, v pour verbose, etc. ;
- le Tag correspond en général au nom de votre classe mais vous pouvez mettre n'importe quelle chaîne de caractères.

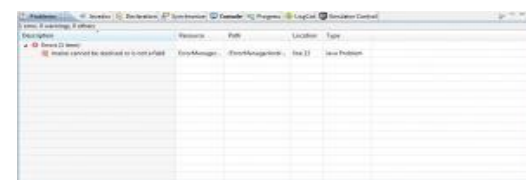
2. Console



La console vous permet entre autres de suivre :

- l'état de compilation de votre application ;
- l'état d'installation de votre application ;
- les warnings ou erreurs lors de la création de l'apk, le téléchargement sur l'émulateur, son installation et le lancement de ce dernier.

3. Problems



L'onglet "**problems**" sert à afficher les erreurs et warnings de programmation, en vous indiquant :

- l'intitulé de l'erreur ;
- le fichier concerné par l'erreur ;
- la ligne ;
- le type du problème.

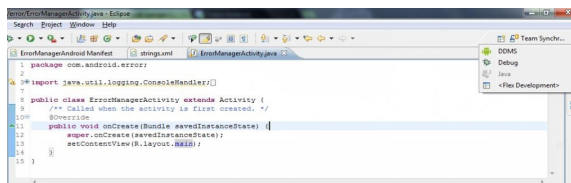
4. Debug

Un des outils les plus utiles quand on programme est de pouvoir poser des **breakpoints**. Pour cela il faut lancer l'application en mode "**Debug As**".

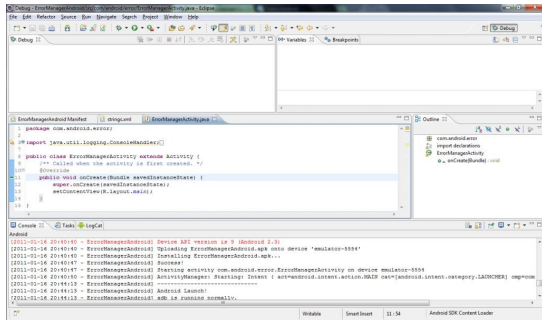
Clic droit sur le projet puis "Debug As -> Android Application". Tous vos breakpoints seront activés.

Quand vous déboguez votre application, n'oubliez pas de passer dans la perspective Debug, pour cela deux solutions.

- En haut à droite de votre IDE, cliquez sur ">>" puis sur "Debug".



Et vous obtiendrez la vue suivante :



- L'onglet "**Variables**" est très utile, il affiche les valeurs des différentes variables.
- L'onglet "**Breakpoints**" affiche la liste de vos breakpoints.

- L'onglet "**Debug**" affiche des informations utiles à votre débogage, par exemple les Threads qui sont en cours d'exécution.

Si la première méthode d'accès au **Debug** ne donne rien, il suffit de cliquer sur "Window -> Open Perspective -> Other" puis sélectionner "**Debug**".

5. Conclusion

J'espère que ce tutoriel vous a aidé. N'hésitez pas à me contacter ou à commenter l'article, si vous rencontrez un problème ou une erreur lors de votre développement.

Retrouvez l'article de Nazim Benbourahla en ligne : [Lien 25](#)

WebMatrix : découverte et prise en main d'un outil de développement Web gratuit, "tout-en-un"

Vous découvrirez au travers de cet article le nouvel environnement de développement Web gratuit et léger WebMatrix, à partir d'exemples simples, nous allons progressivement prendre en main la plate-forme et explorer un ensemble riche d'outils qu'elle met à notre disposition.

1. Introduction

WebMatrix est un environnement de développement Web robuste, léger (15 Mo), efficace et surtout gratuit développé par Microsoft. Il permet aux développeurs Web de créer et gérer des applications Web sur la plate-forme Windows, tout en restant compatible avec les produits Microsoft Visual Studio, SQL Server ou encore PHP sur Windows.

WebMatrix est un outil de développement unique : il vous donne la possibilité d'écrire, modifier et publier des sites Web avec une facilité déconcertante et prend à la fois en charge les langages ASP.NET et PHP, et intègre également la coloration syntaxique et l'IntelliSense pour ces langages.

L'outil est spécialement adapté pour les étudiants, les débutants et les personnes cherchant une solution simple et facile permettant la création d'un site Web sans toute fois avoir besoin de maîtriser l'architecture complexe des technologies Web .NET. WebMatrix minimise le nombre de concepts qu'un débutant en développement a besoin de savoir pour mettre sur pied des sites Web simples.

Vous découvrirez, tout au long de cet article, l'outil de développement Web WebMatrix, ses richesses graphiques et fonctionnelles. Cet article permettra également une prise en main de l'environnement pour créer un site ASP.NET simple pouvant consommer les données d'une base de données SQL Server Compact et pour créer et exécuter un site Web PHP.

2. Description de WebMatrix

WebMatrix regroupe au sein d'un seul outil, les plates-formes et ressources dont les développeurs ont besoin pour créer, exécuter et publier rapidement un site Web à savoir :

- le serveur Web IIS 7 Express, qui est un serveur Web léger simple à installer, pouvant fonctionner avec toutes les versions de Windows et totalement compatible avec IIS 7 ;
- SQL Server Compact Edition 4.0, qui est un gestionnaire de base de données léger, gratuit et simple d'utilisation pouvant être embarqué dans vos applications ASP.NET et migré facilement vers SQL Server ;
- une galerie Web pouvant se connecter au « **Web Application Gallery** » de Microsoft pour proposer aux développeurs une vaste collection de CMS et applications open source populaires comme WordPress, Dupral, Joomla ou encore DotNetNuke, pouvant être installés, édités et

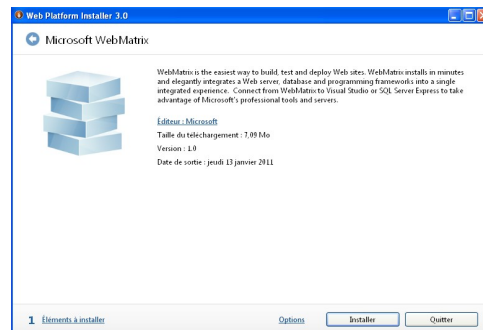
publiés directement à partir de WebMatrix ;

- **Web Deploy** permet d'automatiser le déploiement et la mise à jour des applications sur serveurs ou chez un hébergeur ;
- et enfin un éditeur léger qui prend en charge les langages HTML, HTML5, CSS, ASP.NET et PHP. On note également la prise en charge du Framework Web de Microsoft ASP.NET MVC ainsi que le support du nouveau moteur de vues Razor.

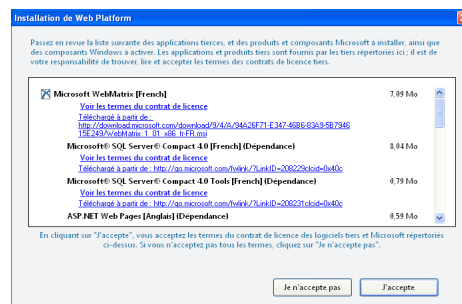
3. Installation de WebMatrix

Pour installer WebMatrix, vous pouvez utiliser le **Web Platform Installer 3.0 (WPI)** qui est téléchargeable gratuitement sur le site de Microsoft. Le lien est fourni en fin d'article.

Sur la page de téléchargement du WPI, cliquez sur **Installer maintenant**, ensuite sur **Exécuter** et enfin sur **Installer**.



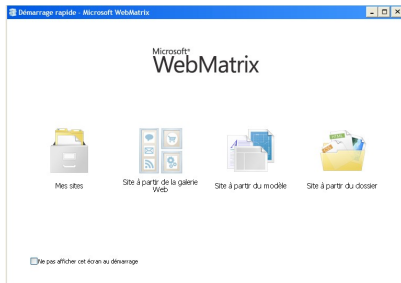
Une fenêtre comportant la liste des applications, produits et composants tierces requis s'affiche. Cliquez sur j'accepte pour lancer l'installation de WebMatrix et des composants liés.



4. Démarrage de WebMatrix

Après avoir installé WebMatrix, vous pouvez lancer l'application à partir du menu **démarrer** -> **Tous les programmes** -> **Microsoft WebMatrix** -> **Microsoft WebMatrix**.

Dès que le chargement de l'application sera achevé, WebMatrix vous proposera un ensemble d'options vous permettant de créer ou accéder rapidement à un site Web.



Par défaut l'éditeur vous propose les choix suivants :

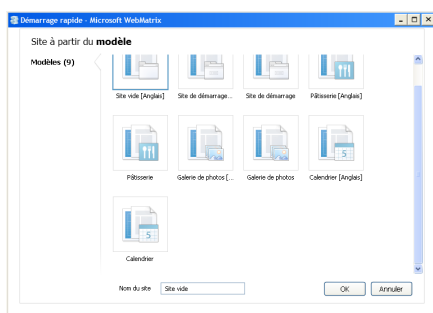
- **Mes Sites**, pour accéder rapidement à vos sites Web existants et les modifier avec WebMatrix ;
- **Site à partir de la galerie Web**, qui vous donne un accès direct à une large gamme de CMS et applications Web open source (DotNetNuke, Dupral, Joomla, WordPress...), pouvant être téléchargés, personnalisés et publiés sans quitter l'éditeur. Pour avoir accès à cette option, vous devez au préalable avoir installé le Web Platform installer et WebDeploy ;
- **Site à partir du modèle**, qui permet de créer un site Web basé sur un modèle prédéfini, ceci permet d'inclure des fichiers et pages Web ayant une certaine structure de base en fonction des différents choix disponibles ;
- **Site à partir du dossier**, qui permet d'ouvrir et éditer un site existant dans un dossier autre que celui dans lequel les sites Web sont par défaut créés avec WebMatrix.

5. Premier site Web avec WebMatrix

Maintenant que nous avons procédé à l'installation et au démarrage de WebMatrix, nous allons créer notre premier site Web basé sur un modèle existant.

Pour cela, dans la fenêtre de "Démarrage rapide", cliquez sur l'option "Site à partir du modèle".

Le programme affiche une liste des différents modèles de site Web disponibles par défaut (Site vide, Site de démarrage, Pâtisserie, Galerie de photos ...).



Sélectionnez le modèle "Site Vide", et dans la zone nom du site, renseignez le nom de votre site Web.

Pour cet exemple introductif, nous allons donner comme nom " MonSiteTest " à notre premier site Web avec WebMatrix.

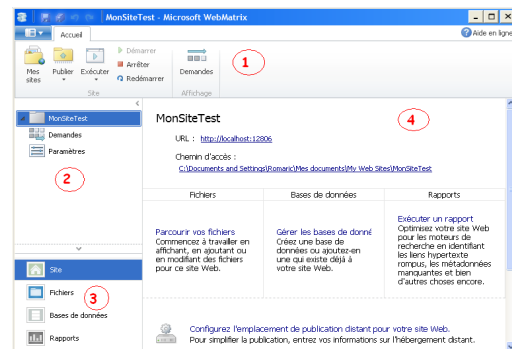
Ensuite cliquez sur le bouton OK pour procéder à la création du site.

Le programme crée le nouveau site et l'affiche dans l'espace de travail de WebMatrix.

5.1. Description de l'espace de travail

Avant de continuer, nous allons présenter brièvement l'interface utilisateur de WebMatrix qui - il faut le dire - est très ergonomique.

WebMatrix vous propose une interface riche mettant à votre disposition tous les outils dont vous avez besoin pour la création et la publication de votre site Web.



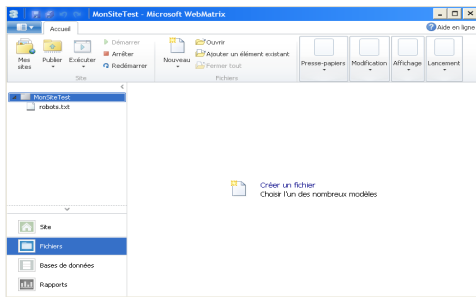
L'interface utilisateur est composée de quatre zones importantes, à savoir : la barre d'outils (1), le panel de navigation (2), le panel de sélection d'espace de travail (3) et la zone de travail (4).

1. **La barre d'outils**: elle contient un ruban "Accueil" façon office 2007 ou 2010. Ce ruban contient les commandes permettant de paramétrer le site (Mes sites, publier un site, exécuter, arrêter...);
2. **Le panel de navigation** : ce panel vous permet de naviguer entre les différentes options disponibles et sélectionner ce qui doit être affiché dans la zone de travail ;
3. **Le panel de sélection d'espace de travail** : cette zone permet de sélectionner l'espace de travail (Site, Fichiers, Base de données, Rapports). En fonction de ce que vous avez sélectionné dans cette zone, les éléments du ruban, du panel de navigation et de la zone de travail sont automatiquement remplacés par ceux adaptés à votre choix ;
4. **La zone de travail** : cette zone est celle où sont affichés les rapports, la zone d'édition des fichiers, de paramétrage du site...

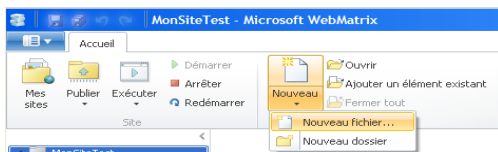
5.2. Création d'une première page Web

Dans le panel de sélection d'espace de travail, sélectionnez l'espace de travail "Fichiers" (cet espace vous permet de créer des fichiers et dossiers de votre site Web).

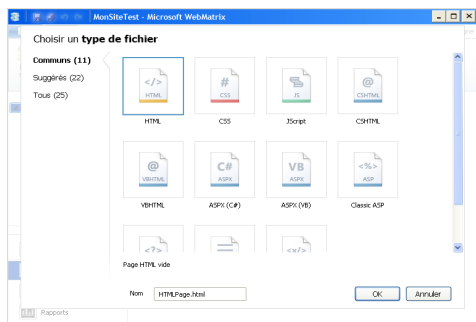
L'espace de travail fichier vous présente la structure des fichiers existants dans votre site dans le panel de navigation.



Dans le ruban de la boîte d'outils, cliquez sur "nouveau", ensuite sur "nouveau fichier" pour procéder à la création d'un nouveau fichier dans notre site.



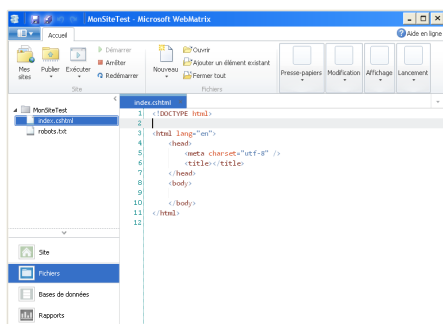
WebMatrix vous affiche une liste des types de fichiers disponibles.



Comme vous pouvez le constater, WebMatrix vous propose plusieurs types de fichiers (html, css, cshtml, vbhtml, php...) qui sont pris en charge par l'éditeur.

Pour notre exemple nous allons utiliser un fichier avec l'extension .CSHTML. Sélectionnez donc dans la liste des fichiers le fichier de type .cshtml et donnez comme nom à votre fichier index.cshtml et cliquez sur OK.

Le fichier index.cshtml est créé avec les lignes de code suivantes par défaut :

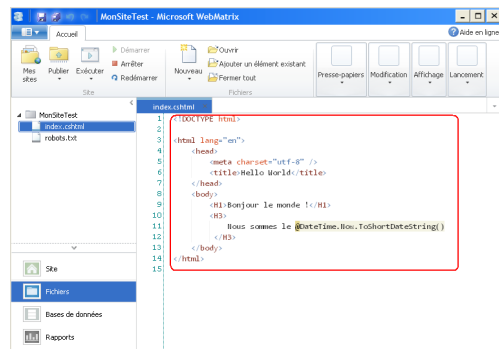


5.3. Qu'est-ce qu'un fichier de type CSHTML ?

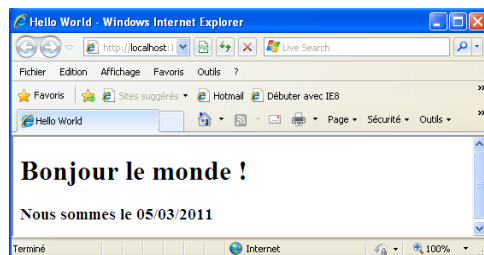
Les fichiers de type cshtml sont des pages Web ASP.NET pouvant contenir du code HTML, JavaScript et CSS tout comme une page HTML normale, sauf qu'ils reposent sur le nouveau moteur de vues Razor et peuvent de ce fait avoir du contenu dynamique.

Les fichiers avec l'extension .vbhtml utilisent également le modèle de vues Razor.

Nous allons, à l'aide de l'éditeur de code de WebMatrix, ajouter quelques lignes de code pour afficher un message et la date du jour dans une page.



Et à l'exécution on obtient ceci :



Vous noterez certainement la syntaxe particulière au niveau de l'appel de la fonction **DateTime**.

```
@DateTime.Now
```

Que cela ne vous effraye pas, c'est juste la syntaxe - assez simple il faut le dire - qu'introduit le nouveau moteur de vues Razor sur lequel repose notre page index.

Ce tutoriel ayant pour principal but de vous faire découvrir WebMatrix et non la syntaxe d'un langage ou d'un moteur de vues, je ne vais pas m'attarder sur la syntaxe de Razor bien que nous l'utilisons dans quelques exemples.

Pour ceux qui éprouvent des difficultés de compréhension de ce tutoriel à cause de cela, je les prie de bien vouloir se référer aux différents tutoriels sur le sujet disponibles sur Internet.

Je tiens quand même à noter que le moteur de vues Razor permet de simplifier énormément les vues en rendant fluide le processus de code et en permettant d'intégrer rapidement du code serveur dans des balises HTML.

6. Utilisation d'une base de données SQL Server Compact

Compact

Passons maintenant à un exemple un peu plus complexe dans lequel nous allons manipuler dans une page Web, des informations provenant d'une base de données. La page que nous allons construire va simplement nous afficher une liste de clients à partir d'une base de données, et une autre page sera également créée pour y insérer des données.

Pour cet exemple, nous allons donc utiliser une base de données SQL Server Compact et la nouvelle syntaxe Razor.

WebMatrix embarque par défaut SQL Server Compact Edition 4.0, avec des outils d'administration de base vous permettant de créer une base de données, créer des tables, des colonnes, ajouter et afficher des données.

SQL Server Compact est particulièrement adapté pour les bases de données de petite taille, et rend la publication d'un site beaucoup plus simple, puisque tous les fichiers de base de données sont directement inclus dans le dossier de votre site Web.

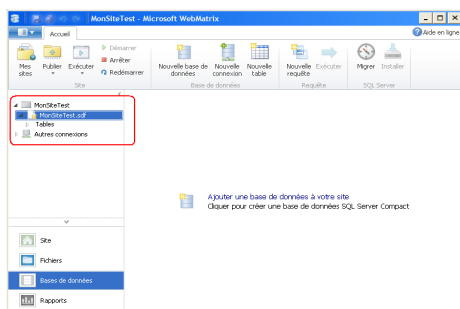
6.1. Création d'une nouvelle base de données

Revenons à notre exemple précédent (MonSiteTest), nous allons ajouter une base de données à notre site.

Pour créer une base de données avec WebMatrix, cliquez sur "Base de données" dans le panel de sélection d'espace de travail pour basculer dans la zone permettant la création et la gestion des bases de données.

Cliquez ensuite sur ajouter une nouvelle base de données. Par défaut, un nouveau dossier App.data est automatiquement ajouté à votre site.

Dans le panel de navigation, un nouveau fichier de base de données (.mdf) est ajouté avec comme nom par défaut celui de votre site (bien évidemment vous pouvez modifier ce nom).



6.2. Ajout d'une nouvelle table

Cliquez sur le bouton nouvelle table dans le ruban pour procéder à la création d'une nouvelle table devant contenir nos clients.

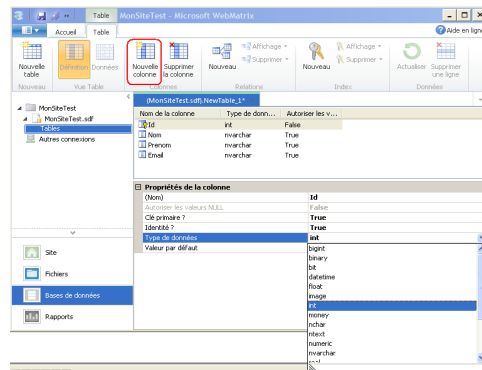
Cliquez ensuite sur nouvelle colonne et créez les différentes colonnes dont nous aurons besoin.

Dans la zone Propriété de la colonne, renseignez le nom, le type et la valeur par défaut.

Pour notre exemple vous allons créer les colonnes : Id de type int, auto-incrément ; Nom de type nvarchar ; Prenom de type nvarchar et Email de type nvarchar.

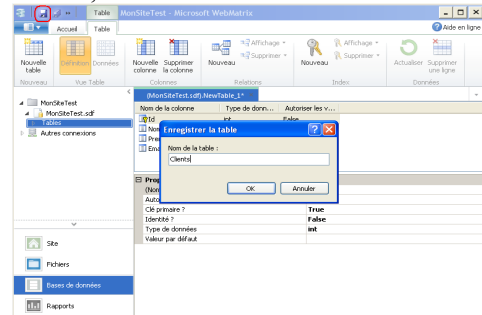
Pour définir le champ Id comme la clé primaire, dans la zone "Clé primaire ?" mettre le champ à " True " et de même pour définir que c'est un champ auto incrément, mettre dans la zone "Identité" le champ à " True " également.

Lorsque vous aurez terminé, vous aurez une table semblable à celle ci-dessous :



Cliquez enfin sur l'icône enregistrer de la boîte d'outils d'accès rapide, et renseignez le nom de la table et cliquez enfin sur OK.

Dans notre cas, on va donner le nom "clients" à la table.



6.3. Ajout des données dans la table

Dans le panel de navigation, sélectionnez la table "clients" et cliquez ensuite sur données dans la barre d'outils pour ajouter ou modifier des données dans la table "clients".

Insérez quelques éléments dans cette table comme l'illustre la capture ci-dessous :

6.4. Affichage des données dans une page Web

Maintenant que nous avons ajouté des données dans notre table "clients", voyons maintenant comment nous pouvons les afficher dans une page Web.

Dans notre site, nous allons ajouter une nouvelle page clients.cshtml. Pour ce cela, "Cliquez" sur Fichiers dans le panel de sélection d'espace de travail, ensuite sur nouveau dans la boîte d'outils. Sélectionnez un fichier de type cshtml et dans la zone nom, saisissez "clients".

Juste avant la balise DOCTYPE, saisissez les lignes de code suivant :

```
@{
    var db = Database.Open("MonSiteTest");
    var sqlreq = "Select * from clients";
}
```

Le signe @ suivi par des accolades, désigne un bloc de code sur plusieurs lignes.

Database est une nouvelle classe qui fournit un ensemble de méthodes pouvant être facilement utilisées pour travailler avec les bases de données.

La méthode **Open()** établit automatiquement une connexion et ouvre la base de données dont le nom du fichier est passé en paramètre.

La seconde ligne de code est juste une requête pour la sélection des données dans la base de données.

Ensuite, dans le body, nous allons écrire le script pour lire les données dans la base de données et les afficher dans un tableau.

Pour cela, nous allons utiliser la méthode **Query** de la classe Database qui prend en paramètre la requête et retourne les champs de la table.

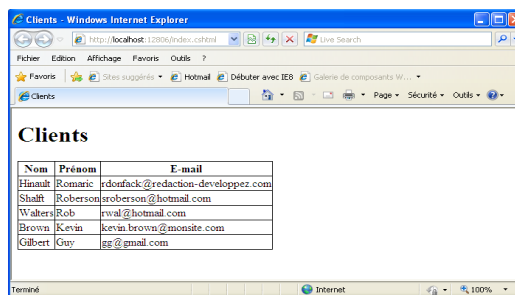
Nous allons également utiliser une boucle **foreach** pour lire les données dans une variable **row** et les afficher dans le tableau.

Le code complet de la page clients.ctlhtml est le suivant :

```
@{
    var db = Database.Open("MonSiteTest");
    var sqlreq = "Select * from clients";
}

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <title>Clients</title>
</head>
<body>
    <H1>Clients</H1>
    <table>
        <tr>
            <th>Nom</th>
            <th>Prénom</th>
            <th>E-mail</th>
        </tr>
        @foreach (var row in db.Query(sqlreq)) {
            <tr>
                <td>@row[1]</td>
                <td>@row[2]</td>
                <td>@row[3]</td>
            </tr>
        }
    </table>
</body>
</html>
```

Et à l'exécution on obtient le résultat suivant :



The screenshot shows a web browser window titled 'Clients - Windows Internet Explorer'. The address bar shows 'http://localhost:12096/index.ctlhtml'. The page content displays a table with the following data:

Nom	Prénom	E-mail
Hinault	Romarc	rdonack@redaction-developpez.com
Shalt	Roberson	roberson@hotmail.com
Walters	Rob	rwal@hotmail.com
Brown	Kevin	kevin.brown@monsite.com
Gilbert	Guy	gg@gmail.com

6.5. Utilisation du WebGrid Helper pour afficher les données

Dans la section précédente, nous avons affiché les données dans une page en utilisant un tableau HTML. Cependant, il existe un moyen beaucoup plus simple d'afficher les données à l'aide d'un nouveau composant ASP.NET : le **WebGrid Helper**.

Le WebGrid Helper retourne un tableau HTML qui affiche les données d'une table. L'assistant prend en charge les options de mise en forme pour créer un moyen de parcourir les données, et pour laisser aux utilisateurs la latitude de trier en cliquant sur un titre de la colonne.

Pour utiliser ce composant, nous allons déclarer une nouvelle variable de type Webgrid comme suit :

```
var grid = new WebGrid(db.Query(sqlreq));
```

Le WebGrid prend en paramètre le résultat d'une instruction de sélection des données dans la BD.

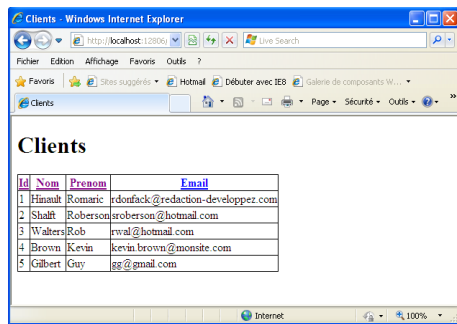
La méthode **@grid.GetHtml()** sera par la suite utilisée dans le body pour retourner une représentation HTML de la grille(tableau HTML).

Le code complet est le suivant :

```
@{
    var db = Database.Open("MonSiteTest");
    var sqlreq = "Select * from clients";
    var grid = new WebGrid(db.Query(sqlreq));
}

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <title>Clients</title>
    <style type="text/css">
        table {border-collapse: collapse;}
        td, th {border: solid 1px; }
    </style>
</head>
<body>
    <h1>Clients</h1>
    @grid.GetHtml()
</body>
</html>
```

Et à l'exécution, on obtient le résultat suivant :



Id	Nom	Prénom	Email
1	Hinault	Romarc	rdonfack@redaction-developpez.com
2	Shaft	Roberson	roberson@hotmail.com
3	Walters	Rob	rval@hotmail.com
4	Brown	Kevin	kevin.brown@monsie.com
5	Gilbert	Guy	gg@gmail.com

6.5.1. Personnaliser un WebGrid

Le WebGrid Helper affiche par défaut toutes les colonnes retournées par la requête. Mais vous pouvez le personnaliser en spécifiant par exemple les données qui seront affichées et dans quel ordre, procéder aux formatages des données (à l'exemple des dates ou des valeurs monétaires).

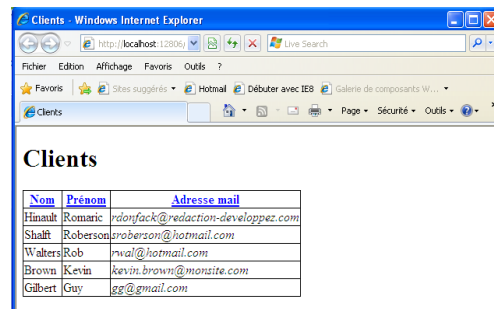
La personnalisation du WebGrid est assez simple, il suffit juste lors de l'appel du Grid.GetHtml, de spécifier les colonnes à afficher, ainsi que le format de celles-ci.

```
@grid.GetHtml(  
    columns: grid.Columns(  
        grid.Column("Nom"),  
        grid.Column("Prénom", "Prénom"),  
        grid.Column("Email", "Adresse  
mail", format: @<i>@item.Email</i>  
    )  
)
```

Le code complet est le suivant :

```
@{  
    var db = Database.Open("MonSiteTest");  
    var sqlreq = "Select * from clients";  
    var grid = new WebGrid(db.Query(sqlreq));  
}  
<!DOCTYPE html>  
<html lang="en">  
    <head>  
        <meta charset="utf-8" />  
        <title>Clients</title>  
        <style type="text/css">  
            table {border-collapse: collapse;}  
            td, th {border: solid 1px; }  
        </style>  
    </head>  
    <body>  
        <h1>Clients</h1>  
        @grid.GetHtml(  
            columns: grid.Columns(  
                grid.Column("Nom"),  
                grid.Column("Prénom", "Prénom"),  
                grid.Column("Email", "Adresse  
mail", format: @<i>@item.Email</i>  
            )  
        )  
    </body>  
</html>
```

Et à la compilation on obtient le résultat suivant :



Nom	Prénom	Adresse mail
Hinault	Romarc	rdonfack@redaction-developpez.com
Shaft	Roberson	roberson@hotmail.com
Walters	Rob	rval@hotmail.com
Brown	Kevin	kevin.brown@monsie.com
Gilbert	Guy	gg@gmail.com

6.5.2. Paginer un WebGrid Helper

Dans le cas où vous avez plusieurs éléments enregistrés dans la base de données, il est possible avec le WebGrid de définir le nombre de lignes qui seront affichées dans la grille.

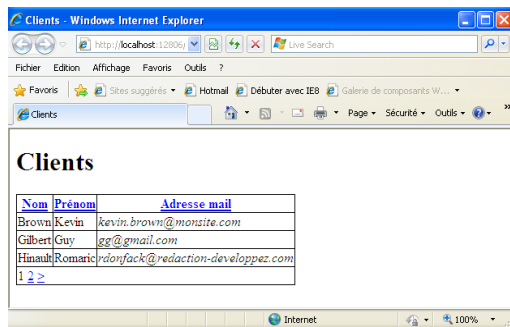
Pour cela, il suffit de surcharger le constructeur du WebGrid en spécifiant la colonne qui sera utilisée pour le tri et le nombre d'éléments à afficher par page.

```
var grid = new WebGrid(source: db.Query(sqlreq),  
    defaultSort: "Nom",  
    rowsPerPage: 3) ;
```

Le code complet est le suivant :

```
@{  
    var db = Database.Open("MonSiteTest");  
    var sqlreq = "Select * from clients";  
    var grid = new WebGrid(source:  
db.Query(sqlreq),  
        defaultSort: "Nom",  
        rowsPerPage: 3) ;  
}  
<!DOCTYPE html>  
<html lang="en">  
    <head>  
        <meta charset="utf-8" />  
        <title>Clients</title>  
        <style type="text/css">  
            table {border-collapse: collapse;}  
            td, th {border: solid 1px; }  
        </style>  
    </head>  
    <body>  
        <h1>Clients</h1>  
        @grid.GetHtml(  
            columns: grid.Columns(  
                grid.Column("Nom"),  
                grid.Column("Prénom", "Prénom"),  
                grid.Column("Email", "Adresse  
mail", format: @<i>@item.Email</i>  
            )  
        )  
    </body>  
</html>
```


Et à la compilation on obtient le résultat suivant :



6.6. Insertion des données à partir d'un formulaire

Dans cette partie, nous verrons comment insérer les données à partir d'une page Web dans une table de notre base de données SQL Server Compact.

Pour cela, dans notre site, nous allons ajouter une nouvelle page nommée **InsertClients.shtml**.

Dans cette page, nous allons ajouter trois zones de saisie pour enregistrer les éléments sur les clients et un script pour vérifier que les données ont été bien saisies.

Après vérification de la saisie, les données seront ensuite enregistrées dans la table clients et l'utilisateur sera automatiquement redirigé vers la page clients.shtml mise à jour.

La méthode **.Execute()** de la classe Database sera utilisée pour l'insertion des données dans la base de données.

La méthode **ModelState.AddError** de la classe **ModelState** sera utilisée pour notifier à la vue qu'il y'a une erreur sur un champ du formulaire.

Le code complet de la page InsertClient.shtml est le suivant (après avoir validé un nouvel enregistrement sans avoir spécifié le champ "Adresse mail") :

```
<!DOCTYPE html>
@{
    var db = Database.Open("MonSiteTest");
    var Nom = Request["Nom"];
    var Prenom = Request["Prenom"];
    var Email = Request["Email"];

    if (IsPost) {

        //vérification du nom
        Nom = Request["Nom"];
        if (Nom.IsEmpty()) {
            ModelState.AddError("Nom", "Le nom est obligatoire.");
        }

        //vérification du prénom
        Prenom = Request["Prenom"];
        if (Prenom.IsEmpty()) {
            ModelState.AddError("Prenom",
                "Le prénom est obligatoire.");
        }
    }
}
```

```
//vérification de l'adresse mail
Email = Request["Email"];
if (Email.IsEmpty()) {
    ModelState.AddError("Email", "L'adresse mail est obligatoire.");
}

if (ModelState.IsValid) {
    var sqlreq = "INSERT INTO clients (Nom, Prenom, Email) " +
        "VALUES (@0, @1, @2)";
    db.Execute(sqlreq, Nom, Prenom, Email);
    // Display the page that lists products.
    Response.Redirect(@Href("~/Clients"));
}
}
```

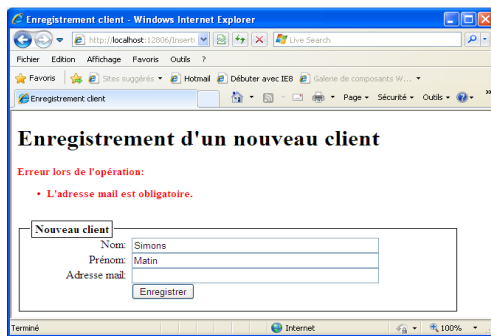
```
<html>
<head>
    <title>Enregistrement client</title>
    <style type="text/css">
        label {float:left; width: 8em; text-align:right;
            margin-right: 0.5em;}
        fieldset {padding: 1em; border: 1px solid; width: 35em;}
        legend {padding: 2px 4px; border: 1px solid; font-weight:bold;}
        .validation-summary-errors {font-weight:bold; color:red; font-size: 11pt;}
    </style>
</head>
<body>
    <h1>Enregistrement d'un nouveau client</h1>

    @Html.ValidationSummary("Erreur lors de l'opération:")

    <form method="post" action="">
        <fieldset>
            <legend>Nouveau client</legend>
            <div>
                <label>Nom:</label>
                <input name="Nom" type="text" size="50" value="@Nom" />
            </div>
            <div>
                <label>Prénom:</label>
                <input name="Prenom" type="text" size="50" value="@Prenom" />
            </div>
            <div>
                <label>Adresse mail:</label>
                <input name="Email" type="text" size="50" value="@Email" />
            </div>
            <div>
                <label>&nbsp;</label>
                <input type="submit" value="Enregistrer" class="submit" />
            </div>
        </fieldset>

    </form>
</body>
</html>
```

Et à la compilation, vous obtenez le résultat suivant :



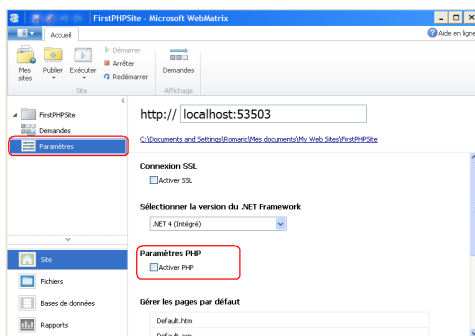
7. WebMatrix et PHP

Comme nous l'avons déjà mentionné plus haut, WebMatrix prend complètement en charge et intègre un éditeur de code avec coloration syntaxique pour le langage PHP. Vous avez la possibilité avec l'éditeur de créer et exécuter des sites Web PHP où de modifier les applications Web existantes dans la Gallery d'applications.

Dans cette section, nous verrons comment créer une application PHP, configurer automatiquement PHP avec le serveur IIS express à partir de WebMatrix et exécuter un site Web.

Pour cela, créez une nouvelle application avec pour nom **FirstPHPSite**.

Dans le panel de sélection d'espace de travail, cliquez sur site, ensuite dans le panel de navigation sur paramètres et dans la zone de travail, dans l'espace "Paramètres PHP", cochez la zone "Activer PHP".



WebMatrix affiche automatiquement les versions de PHP qui sont prises en charge en vous donnant la possibilité de les installer si n'est pas encore fait.



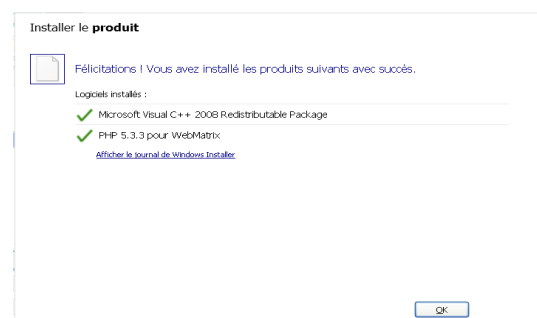
Après sélection de la version que vous souhaitez utiliser, le programme procède automatiquement au téléchargement

de celui-ci et des composants requis.



La version de PHP téléchargée est celle recommandée pour l'exécution de PHP sur Windows via la prise en charge FastCGI fournie dans IIS.

Lorsque le téléchargement est achevé, vous aurez la fenêtre suivante :

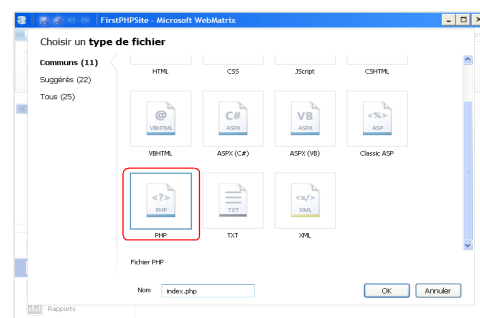


Pendant cette opération, WebMatrix procède à une configuration automatique d'IIS Express pour que celui-ci puisse exécuter votre site Web PHP.

Maintenant nous allons ajouter une page index.php à notre site.

Dans le panel de sélection d'espace de travail, cliquez sur fichier et ensuite sur nouveau fichier dans la barre d'outils, puis sur nouveau.

Dans la fenêtre de sélection du type de fichier, sélectionnez le type PHP et donnez au fichier le nom index.php.



Ajoutez les lignes de code suivantes à ce fichier :

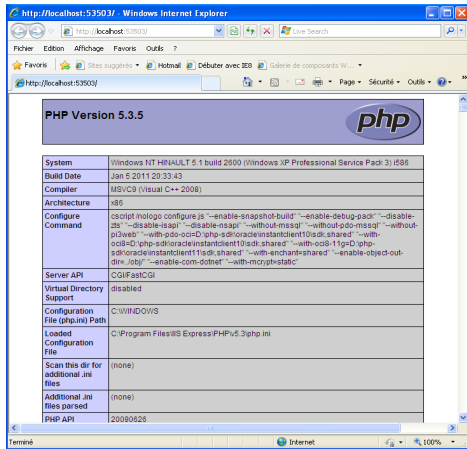
```
<code>!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <title></title>
  </head>
  <body>
```

```

<?php
phpinfo();
?>
</body>
</html>

```

Et à la compilation vous obtenez le résultat suivant :



Maintenant vous pouvez commencer à écrire vos applications PHP à l'aide de WebMatrix.

8. Conclusion partielle

Avec cette version de WebMatrix, vous avez à votre disposition un environnement de développement Web mature, riche en fonctionnalités et très facile d'utilisation. Que vous soyez étudiant, débutant en développement ou que vous souhaitiez mettre sur pied de simples sites Web, WebMatrix est fait pour vous. Avec un minimum de connaissance en développement .NET, il vous offre la possibilité de créer des sites Web riches.

9. Liens

Télécharger WebMatrix : [Lien 26](#)

Blog de Scott Guthrie : [Lien 27](#)

Les nouveautés d'ASP.NET MVC 3 : [Lien 28](#)

Retrouvez l'article de Hinault Romaric DONFACK en ligne : [Lien 29](#)

Développement Web

Les derniers tutoriels et articles

État des lieux de l'accessibilité de HTML5

Après avoir affirmé à Paris Web 2010 que HTML5 était utilisable immédiatement en production ([Lien 30](#)), un expert en accessibilité m'a repris en disant qu'il était dangereux de prétendre que HTML5 était accessible (j'en parlais au futur cela dit). Dans le cadre d'un gros projet autour de HTML5, j'ai depuis fait pas mal de recherches, ce qui m'a permis de mieux comprendre son intervention.

Je sais qu'il est dangereux de parler en dehors de son domaine d'expertise, mais il faut bien qu'un développeur Web mette les pieds dans le plat et à tout le moins provoque le débat. Autant vous dire que si vous vous y connaissez en accessibilité, je prends tout ajout ou correction.

Enfin, mes conclusions sont mitigées et il se pourrait même que je revienne sur ce que j'affirmais à l'époque.

L'article original et les commentaires peuvent être lus sur le site BrainCracking ([Lien 31](#)) : Etat des lieux de l'accessibilité de HTML5 ([Lien 32](#)).

1. Quel HTML5 ?

Le terme commençant à être flou, il vaut mieux le préciser à chaque fois qu'on l'évoque : pour une fois sur ce blog, le HTML5 dont nous parlerons dans cet article représente uniquement la spécification adoubee par le W3C ([Lien 33](#)), celle qui correspond à l'évolution de HTML4 avec les nouvelles balises, le nouvel algorithme de parsing, canvas et le multimédia.

2. Les nouvelles balises

2.1. Leur apport

Les balises comme article, time, header, nav ou figure apportent de la sémantique à votre page. Cela signifie concrètement plusieurs choses :

- une meilleure **lisibilité du code**, ce qui est important pour ceux qui l'écrivent et facilite la maintenance ;
- une (future) exploitation facilitée pour les tiers : Readability (application de lecture, service de rétribution des auteurs de blogs : [Lien 34](#)) se base sur ces nouvelles balises (ou sur microformat) pour tirer les informations des articles des pages. Il est difficile de trouver d'autres exemples pour le moment ;
- un (futur) meilleur référencement : la position officielle de Google sur le sujet est qu'il adapte ses algorithmes à toutes les pratiques émergentes. Si beaucoup de sites utilisent article, Google finira par l'interpréter ;
- une (future) meilleure accessibilité : les navigateurs vont rajouter automatiquement ([Lien 33](#)) des rôles ARIA sur certaines balises. Ce tableau ([Lien 35](#)) vous montre le support actuel, il est aujourd'hui très limité : Firefox 4 pour certaines balises, Opéra pour les formulaires sont leaders.

Bref, les apports immédiats pour tous les sites ne sont pas évidents. Il y aura tout de même un bonus pour ceux qui jouent l'innovation et y passent avant les concurrents

lorsque ces lecteurs de code source (services, moteurs de recherche, navigateurs) exploiteront cette mine d'informations.

2.2. Les problèmes

Le reproche principal fait aux nouvelles balises, c'est que IE6 à 8 font disparaître du DOM les éléments qu'ils ne connaissent pas et donc ne les présentent pas aux technologies d'assistance. Le moyen de contourner cela est connu ([Lien 36](#)), facile à mettre en place et *bullet-proof* mais il induit une **dépendance envers JavaScript**. Les derniers chiffres français pris sur la page d'accueil de Yahoo! ([Lien 37](#)) font état de 1.5 % d'utilisateurs dans cet état. La part de marché de IE est d'environ 50 %, cela vous laisse avec 0.7 % d'utilisateurs que cela peut gêner parmi vos visiteurs handicapés.

À ma connaissance, il y a deux autres choses qui ont donné une mauvaise réputation à ces nouveaux éléments :

- une combinaison de certaines versions de JAWS et de Firefox avait un bogue qui faisait **disparaître le contenu** situé dans une balise header. En plus de la perte brute de contenu, selon où elle est placée, header peut contenir des titres ou des liens, deux éléments fondamentaux pour la navigation avec les technologies d'assistance ;
- une autre combinaison, celle des versions de Windows-Eyes et d'Internet Explorer, n'arrivait pas à lister les liens à l'intérieur d'éléments HTML5 avec une balise role. C'est très spécifique mais là aussi il y a perte d'éléments de navigation fondamentaux.

Doit-on changer sa manière de coder parce que deux logiciels boguent ? Historiquement les développeurs Web ont toujours répondu oui à cette question, en changeant leur code pour s'adapter aux bogues de IE par exemple. Sauf que IE est officiellement supporté par tous les sites Web alors que ce n'est pas le cas pour ces deux logiciels. Même si ces bogues ont été corrigés, **ils restent majeurs** (perte de contenu). Je n'ai pas de chiffre sur le taux de pénétration de ces versions boguées, mais les seuls chiffres

publics que j'ai pu trouver montrent que 25 % de ces utilisateurs n'ont pas mis à jour leur version après un an ([Lien 38](#)), ce qui signifie qu'il faut prendre en compte ces bogues durant plusieurs années.

2.3. En conclusion

L'introduction des nouvelles balises est surtout compromise par des bogues majeurs de certains logiciels importants et votre **sentiment** sur la dépendance à JavaScript (que vous devriez d'ailleurs mesurer sur votre site). Elle est à mettre en balance avec les avantages de ces balises, qui eux ne sont pas immédiats.

3. Les zones de navigation

L'introduction des balises nav, header, footer et aside est très intéressante car elle permet au navigateur de fournir aux technologies d'assistance une carte de la page. Jusqu'ici les utilisateurs de ces technologies naviguaient principalement avec la liste des titres et la liste des liens d'une page, ils pourraient également avoir accès automatiquement aux zones classiques présentes sur les sites Web.

Première ombre au tableau, HTML5 définit ([Lien 33](#)) un mapping strict entre certaines balises et des rôles de zone ARIA, mais sur ces zones là il n'y a pas de rôle par défaut. Il y a donc contradiction entre la spécification et les rôles que les navigateurs devraient automatiquement implémenter (`<nav role="navigation">`, `<header role="banner">`, `<footer role="contentinfo">`, `<aside role="complementary">`). Qui dit contradiction dit mésentente possible entre navigateurs.

Ensuite vient le support : la traduction automatique des balises en rôle de zone ARIA par les navigateurs arrive à peine, Firefox 4 étant pour le moment seul. Par contre ARIA est bien supporté par les versions récentes des lecteurs d'écran. Sauf que comme pour les navigateurs, les clients de ces logiciels mettent un certain temps avant de passer aux nouvelles versions.

Enfin, ces rôles et ces balises sont censés permettre la disparition des liens d'échappement, bonne pratique d'accessibilité reconnue. Mais ça serait oublier ceux qui n'utilisent pas ces logiciels, mais qui naviguent tout de même au clavier (choix, accident de souris, mauvais matériel, certains mobiles...). A ma connaissance, les navigateurs n'ont pas prévu de fonctionnalité pour ces utilisateurs, en leur mettant à disposition des raccourcis vers ces zones.

3.1. En conclusion

La technique du lien d'évitement restera très longtemps un moyen de navigation plus universel. Si vous ne la pratiquez pas, utiliser les nouvelles balises maintenant apportera naturellement de l'accessibilité au fur et à mesure du déploiement des nouvelles versions de navigateurs et de technologie d'assistance.

4. Le multimédia

Les éléments audio et video natifs ont fait rêver un peu tout le monde et sur le papier règlent d'épineux problèmes. En pratique c'est beaucoup plus compliqué (sur mon

projet, nous avons jugé que les implémentations navigateur ne valaient pas encore Flash) et l'accessibilité n'est pas en reste. Les implémentations des navigateurs concernant la navigation clavier sont variées, parfois boguées et parfois inexistantes.

Les sous-titres ne sont implémentés nativement nulle part et les spécifications ont vraiment beaucoup bougé à ce sujet. Difficile de savoir si le consensus actuel autour de l'élément track et le format WebSrt va rester.

Comme vous devez de toute manière fournir une lecture de la vidéo en Flash, qui lui est naturellement encore moins accessible (il pourrait, mais les *flasheurs* ne le font pas), votre seule option est de coder vous-même un lecteur vidéo accessible, en sortant les contrôles du lecteur (natif ou Flash) pour en faire des éléments HTML marqués avec ARIA ([Lien 39](#)) et d'implémenter votre propre solution de sous-titrage en JavaScript. Mais vous sacrifiez l'option fullscreen, ce qui est rarement toléré.

4.1. En conclusion

Les balises audio et video sont vraiment trop jeunes concernant l'accessibilité et il n'y a pas de gros bénéfices à en tirer sur les navigateurs de bureau. Flash étant pire, vous devez améliorer vous-même les choses en codant un lecteur accessible ou en espérant en trouver un dans cette longue liste : [Lien 40](#).

5. Les titres

HTML5 définit un algorithme pour déterminer le vrai niveau des Hx d'une page ([Lien 41](#)). Concrètement si vous mettez un h1 dans une balise section ou article qui n'est elle-même pas enfant d'une autre balise sectionnante, alors votre h1 est en fait un h2. C'est très pratique pour ceux qui développent des sites Web de façon modulaire : on ne s'occupe que de la hiérarchie interne d'un bout de page, sans avoir à se demander quel est le nombre de niveaux de titre déjà introduit. Pour voir cet algorithme en action, vous pouvez utiliser l'outil HTML5 outliner : [Lien 42](#).

En quoi est-ce important ? Les titres sont la première manière de naviguer sur une page lorsque l'on utilise un lecteur d'écran et selon le moteur HTML utilisé, la hiérarchie interprétée ne sera pas la même et vous ne pouvez rien y faire. Honnêtement si vous n'avez pas prêté attention à la hiérarchie globale des titres jusqu'à présent, HTML5 vous aidera à mieux vous organiser module par module et à obtenir au final quelque chose de cohérent au moins sur les nouveaux navigateurs. Si vous aviez une belle hiérarchie, alors pour la conserver sur tous les navigateurs, il vous faudra éviter les balises sectionnantes comme article, section ou aside.

Cela dit, une hiérarchie bien maîtrisée de titres n'est réellement importante que si la page comporte beaucoup de titres, par exemple sur des sites à fort contenu textuel (longs articles de journaux, encyclopédie...). La plupart des sites se contentent d'une vingtaine de titres de niveau 1 et 2.

5.1. En conclusion

Si vous ne gérez pas votre hiérarchie parfaitement au

niveau de la page (site très modulaire, maintenu par plusieurs personnes, ou par méconnaissance), autant passer directement à la logique HTML5, vous devrez vivre avec une hiérarchie de titres différente selon le navigateur. Si vous aviez une belle hiérarchie et beaucoup de titres, alors pour la conserver sur tous les navigateurs, il vous faudra éviter les balises sectionnantes comme article, section ou aside.

6. Canvas

Canvas est effectivement un trou noir d'où rien ne sort et ironiquement, même Flash peut être plus accessible que cette balise (encore une fois, si le développeur Flash a fourni un effort supplémentaire). Seul IE9 permet d'accéder à *shadow DOM*, mais en supposant que tous les navigateurs fassent de même, la question de l'intérêt se pose : pourquoi accéder à des paquets de pixels ? Si vous l'utilisez pour obtenir des effets graphiques décoratifs, un contenu alternatif textuel peut suffire. Si vous l'utilisez pour la navigation, vous vous êtes probablement trompé de technologie. Si votre application repose de manière justifiée sur cette technologie, comme pour un jeu, alors il me semble de toute façon impossible de rendre accessible ce type d'application très graphique.

6.1. En conclusion

Ressortez donc les bonnes pratiques, fournissez un contenu alternatif si c'est possible et évaluez bien vos besoins : SVG / VML ou une navigation scriptée peuvent suffire.

7. Passer à ARIA

Ma conclusion personnelle est qu'il faut rajouter ARIA et ses multiples rôles à votre page HTML5. Voici ce que j'utilise occasionnellement pour tester l'ajout d'ARIA ou pour vérifier l'accessibilité d'une page. Pour tester ARIA, il vous faut un Windows, un Firefox et un Internet Explorer 8 qui sont les deux navigateurs les plus utilisés par les clients des lecteurs d'écran. Ce sont également les deux navigateurs qui supportent le mieux ARIA. En lecteur d'écran JAWS par Freedom Scientific ([Lien 43](#)) est

clairement le leader du marché, suivi par Window-Eyes de GW Micro ([Lien 44](#)). Ils offrent tous deux des versions d'essai gratuites dont la limite est que vous devez redémarrer Windows toutes les 40 minutes.

Astuce : utilisez des machines virtuelles et la fonctionnalité snapshot pour vous éviter ces redémarrages fastidieux. Vous pouvez également utiliser un lecteur d'écran open-source : NVDA ([Lien 45](#)) ainsi que la toolbar Juicy Studio Accessibility Toolbar ([Lien 46](#)) pour Firefox qui permet de vérifier ses zones pendant le développement.

8. Conclusion

Vaste sujet que l'accessibilité de HTML5 et j'imagine (j'espère!) que des experts en accessibilité passeront sur cet article pour me corriger ou rajouter d'autres difficultés. En conclusion :

- il me faut l'admettre, l'introduction des nouvelles balises peut causer des bogues majeurs ou gêner des logiciels de lecture d'écran. Ces bogues et cette gêne de la hiérarchie sont là pour plusieurs années tandis que les avantages des nouvelles balises ne sont pas encore massivement arrivés. À vous de voir ;
- l'utilisation des balises video et audio est prématurée à tous les points de vue ;
- il va falloir encore du temps avant que HTML5 ne vous apporte plus d'accessibilité que ARIA. Soit vous passez à ARIA immédiatement, soit vous pariez sur le futur.

Si le ton négatif de ce billet vous fait conclure qu'il ne faut pas utiliser HTML5, alors je vous invite à me relire : j'ai principalement **listé les problèmes** afin que vous sachiez à quoi vous vous engagez en restant sur les techniques actuelles ou en partant sur certaines nouveautés de HTML5. HTML5 au sens des applications Web avec toutes ses nouvelles APIs JS ne sont ici pas concernées.

Retrouvez l'article de Jean-Pierre Vincent en ligne : [Lien 47](#)

Créer des listes ordonnées attrayantes en CSS

Cet article est la traduction de : Sexy Ordered Lists with CSS ([Lien 48](#)).

J'ai récemment travaillé sur un site pour lequel j'ai dû créer des styles spéciaux pour les listes ordonnées, j'ai donc pensé que cela pourrait vous servir pour vos propres projets.

Le problème pour beaucoup est de comprendre comment séparer les styles de la numérotation de ceux du contenu de la liste.

Il semblerait plus facile d'intégrer la numérotation dans le contenu d'une liste non ordonnée et de lui affecter un style, mais ce serait contourner le problème au lieu de le résoudre !

Dans ce tutoriel, je vais vous montrer une solution pour réussir cette mise en forme.

Vous pouvez déjà voir le résultat attendu : [Lien 49](#).

1. Le code HTML

Nous allons d'abord créer une liste ordonnée avec pour chaque élément un titre et un paragraphe

```
<ol class="steps">
  <li>
    <h2>heading</h2>
```

```
<p>paragraph</p>
  </li>
  <li>
    <h2>heading</h2>
    <p>paragraph</p>
  </li>
</ol>
```

2. Le code CSS

Nous commençons par donner un style à la liste elle-même. Notez que nous précisons des styles de police pour les éléments de liste, ce seront ceux appliqués à la numérotation

```
ol.steps {
    margin: 20px 0;
    background: url(ul_bg_repeat.gif) repeat-y; /*--Image de fond de la numérotation--*/
    padding: 0 0 0 35px; /*--Décalage de la numérotation--*/
    border: 1px solid #111;
}
ol.steps li {
    margin: 0;
    padding: 15px 15px;
    color: #cbff78;
    font-size: 1.7em;
    font-weight: bold;
    /*--Le style biseauté est créé avec les styles suivants appliqués aux bordures--*/
    border-top: 1px solid #000;
    border-bottom: 1px solid #353535;
    border-right: 1px solid #333;
    border-left: 1px solid #151515;
    background: #222;
}
```

En créant le style biseauté avec les couleurs de bordures, un pixel supplémentaire apparaît en haut et en bas de la liste.

Nous pouvons remédier à cela de deux façons, mais d'abord, nous ajoutons une classe spécifique pour le premier et le dernier élément de la liste.

```
ol.steps li.first { border-top: 1px solid #353535; }
ol.steps li.last { border-bottom: none; }
```

2.1. Méthode manuelle

Il nous suffit d'ajouter le nom de la classe manuellement dans le code HTML de la liste

```
<li class="first"><!--Contenu--></li>
<li><!--Contenu--></li>
<li class="last"><!--Contenu--></li>
```

2.2. Méthode avec jQuery

Utilisons jQuery pour ajouter les noms de classe : [Lien 50](#).

```
<script type="text/javascript">
$(document).ready(function() {
    $("ol.steps li:first").addClass("first");
    $("ol.steps li:last").addClass("last");
});
</script>
```

Maintenant que nous en avons fini avec la liste, nous pouvons appliquer des styles aux balises de titre et de paragraphe pour annuler les styles par défaut de la liste

```
ol.steps li h2 {
    font-size: 0.9em;
    padding: 5px 0;
    margin-bottom: 10px;
    border-bottom: 1px dashed #333;
    color: #fff;
}
ol.steps li p {
    color: #ccc;
    font-size: 0.7em;
    font-weight: normal;
    line-height: 1.6em;
}
```

3. Conclusion

Comme vous pouvez le voir, les listes ordonnées ne sont pas forcément destinées à être austères.

Bien entendu, si vous avez d'autres techniques pour styliser vos listes, proposez-les dans les commentaires de ce tutoriel.

Voir un exemple de ce code : [Lien 49](#).

Retrouvez l'article de Soh Tanaka traduit par Didier Mouronval en ligne : [Lien 51](#)

Le tutoriel SPARQL

L'objectif de ce tutoriel est de donner un cours rapide sur SPARQL. Il couvre toutes les fonctionnalités majeures du langage de requête à travers des exemples, mais ne vise pas à être complet.

Si vous cherchez une introduction à SPARQL et Jena, regardez Recherche dans les données RDF avec SPARQL : [Lien 52](#).

SPARQL est un langage de requêtes ([Lien 53](#)) et un protocole ([Lien 54](#)) pour l'accès RDF, conçu par le groupe de travail du W3C RDF Data Access ([Lien 55](#)).

En tant que tel, SPARQL est orienté données, en ce sens qu'il n'effectue des recherches que sur des informations contenues dans des modèles ; il n'y a pas d'inférence dans le langage de requête lui-même. Évidemment, le modèle Jena peut être « intelligent », en ce sens qu'il fournit l'impression que certains triplets existent en les créant à la demande, y compris le raisonnement OWL. SPARQL ne fait rien d'autre que prendre la description de ce que l'application veut sous la forme d'une requête et retourne cette information sous la forme d'un ensemble de données liées ou d'un graphe RDF.

1. L'article original

Cet article est la traduction de SPARQL Tutorial : [Lien 56](#).

2. Préliminaires : les données

Tout d'abord, il faut être clair sur les données sur lesquelles seront exécutées les requêtes. SPARQL effectue des recherches sur des graphes RDF. Un graphe RDF est un ensemble de triplets. Jena appelle les graphes « modèles » et les triplets « déclarations », car c'est comme ça qu'on les appelait au moment où l'API Jena a été conçue.

Il est important de remarquer que ce sont les triplets qui ont de l'importance, pas la sérialisation. La sérialisation est juste une manière de coucher par écrit les triplets. RDF/XML est la recommandation du RDF, mais il peut être difficile de voir les triplets dans cette forme, car il y a plusieurs méthodes pour encoder le même graphe. Dans ce tutoriel, on utilisera une approche de la sérialisation plus proche des triplets, Turtle ([Lien 57](#)) (voir aussi le langage N3 décrit dans W3C semantic web primer : [Lien 58](#)).

On va commencer avec les données simples de vc-db-1-rdf ([Lien 59](#)) : ce fichier contient un graphe RDF pour un certain nombre de descriptions au format VCARD de gens. VCARD est décrit dans la RFC2426 ([Lien 60](#)) et le procédé de traduction RDF est décrit dans la note du W3C Representing vCard Objects in RDF : [Lien 61](#). La base de données d'exemple ne contient que quelques informations sur les noms.

Graphiquement, cela ressemble à ceci :



Sous la forme de triplets, plutôt à ceci :

```
@prefix vCard: <http://www.w3.org/2001/vcard-rdf/3.0#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix : <#> .<http://somewhere/MattJones/>
vCard:FN "Matt Jones" ;
vCard:N [ vCard:Family
          "Jones" ;
          vCard:Given
          "Matthew"
        ] .<http://somewhere/RebeccaSmith/>
vCard:FN "Becky Smith" ;
vCard:N [ vCard:Family
          "Smith" ;
          vCard:Given
          "Rebecca"
        ] .<http://somewhere/JohnSmith/>
vCard:FN "John Smith" ;
vCard:N [ vCard:Family
          "Smith" ;
          vCard:Given
          "John"
        ] .<http://somewhere/SarahJones/>
vCard:FN "Sarah Jones" ;
vCard:N [ vCard:Family
          "Jones" ;
          vCard:Given
```

```
] . "Sarah"
```

Ou sous une forme de triplets plus explicites :

```
@prefix vCard: <http://www.w3.org/2001/vcard-rdf/3.0#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .<http://somewhere/MattJones/>
vCard:FN "Matt Jones" .
<http://somewhere/MattJones/> vCard:N _:b0 .
_:b0 vCard:Family "Jones" .
_:b0 vCard:Given "Matthew" .
<http://somewhere/RebeccaSmith/> vCard:FN "Becky Smith" .
<http://somewhere/RebeccaSmith/> vCard:N _:b1 .
_:b1 vCard:Family "Smith" .
_:b1 vCard:Given "Rebecca"
.<http://somewhere/JohnSmith/> vCard:FN "John Smith" .
<http://somewhere/JohnSmith/> vCard:N _:b2 .
_:b2 vCard:Family "Smith" .
_:b2 vCard:Given "John"
.<http://somewhere/SarahJones/> vCard:FN "Sarah Jones" .
<http://somewhere/SarahJones/> vCard:N _:b3 .
_:b3 vCard:Family "Jones" .
_:b3 vCard:Given "Sarah" .
```

Il est important de remarquer qu'il s'agit de représentations du même graphe RDF et que les triplets du graphe n'ont pas d'ordre particulier. Ils sont juste écrits en groupes liés ci-dessus pour qu'un humain puisse les lire, la machine n'y prête pas attention.

3. Exécution d'une requête simple

Dans cette section, on va jeter un œil à une première requête, simple, puis on va l'exécuter avec ARQ.

3.1. Le hello world des requêtes

Le fichier q1.rq ([Lien 62](#)) contient la requête suivante :

```
SELECT ?x
WHERE { ?x <http://www.w3.org/2001/vcard-rdf/3.0#FN> "John Smith" }
```

En exécutant cette requête avec l'application en ligne de commande, on obtient :

```
-----
| x |
| <http://somewhere/JohnSmith/> |
-----
```

Ceci fonctionne par comparaison des motifs des triplets dans la clause WHERE avec les triplets du graphe RDF. Le prédicat et l'objet des triplets sont des valeurs fixées, les comparer avec un motif ne ressortira que les triplets avec ces valeurs. Le sujet est une variable et il n'y a pas d'autre restriction sur la variable. Le motif correspond à n'importe quel triplet avec ces valeurs d'objet et de prédicat et les fait correspondre avec les solutions pour x.

L'item entouré de < > est une URI (précisément, une IRI) et l'item entre "" est une valeur littérale. Comme pour Turtle,

N3 ou N-triplets, les littérales typées sont entourées de ^^ et des langues peuvent être ajoutées avec @.

?x est une variable appelée x. Le ? n'est pas une partie du nom, c'est pourquoi il n'apparaît pas dans la sortie.

Il n'y a qu'une seule correspondance. La requête la retourne dans la variable de requête x. La sortie montrée a été obtenue en utilisant une des applications en ligne de commande d'ARQ.

3.2. Exécuter la requête

Il y a des scripts d'assistance ([Lien 63](#)) dans les répertoires /bat et /bin d'ARQ. Ils pourraient ne pas être présents dans la distribution de Jena.

3.2.1. Préparation sous Windows

Il faut définir la variable d'environnement ARQROOT pour qu'elle pointe sur l'emplacement de la distribution d'ARQ.

```
set ARQROOT=c:\MyProjects\ARQ
```

La distribution est généralement installée dans un dossier dont le nom contient le numéro de version.

Dans le répertoire d'ARQ, exécuter :

```
bat\sparql.bat --data=doc\Tutorial\vc-db-1.rdf
--query=doc\Tutorial\q1.rq
```

On peut simplement mettre le répertoire /bat dans le PATH ou les déplacer, ils dépendent tous de ARQROOT.

3.2.2. Scripts Bash pour Unix/Linux/Cygwin

Il faut définir la variable d'environnement ARQROOT pour qu'elle pointe sur l'emplacement de la distribution d'ARQ.

```
export ARQROOT=$HOME/MyProjects/ARQ
```

La distribution est généralement installée dans un dossier contenant la version dans son nom.

Dans le répertoire d'ARQ, exécuter :

```
bin\sparql --data=doc/Tutorial/vc-db-1.rdf
--query=doc/Tutorial/q1.rq
```

On peut simplement mettre le répertoire /bat dans le PATH ou le déplacer, il dépend de ARQROOT.

Cygwin est un environnement de type Unix pour Windows : [Lien 64](#).

3.2.3. En utilisant directement les applications Java en ligne de commande

On doit alors définir le classpath pour qu'il comprenne tous les fichiers jar du dossier /lib.

Par exemple, sous Windows :

```
ARQdir\lib\antlr-2.7.5.jar;ARQdir\lib\arq-extra.jar;ARQdir\lib\arq.jar;ARQdir\lib\commons-logging-1.1.jar;ARQdir\lib\concurrent.jar;ARQdir\lib\icu4j_3_4.jar;ARQdir\lib\iri.jar;ARQdir\lib\jena.jar;ARQdir\lib\jenatest.jar;ARQdir\lib\json.jar;ARQdir\lib\junit.jar;ARQdir\lib\log4j-1.2.12.jar;ARQdir\lib\lucene-core-2.2.0.jar;ARQdir\lib\stax-api-1.0.jar;ARQdir\lib\wstx-asl-3.0.0.jar;ARQdir\lib\xercesImpl.jar;ARQdir\lib\xml-apis.jar
```

Avec ARQdir le répertoire d'extraction d'ARQ. Tout doit être sur une seule ligne.

Le nom des fichiers jar évolue avec le temps et de nouveaux apparaissent - il suffit de vérifier cela dans la version d'ARQ.

Les commandes elles-mêmes sont dans le paquet arq.

4. Motifs de base

Cette section couvre les bases des motifs et des solutions, les blocs principaux dans la construction de requêtes SPARQL.

4.1. Solutions

Les solutions d'une requête sont l'ensemble des paires d'un nom de variable avec une valeur. Une requête SELECT expose directement les solutions (après tri, limite ou décalage) telles que présentes dans l'ensemble de résultats. D'autres formes de requêtes utilisent ces solutions pour réaliser un graphe. La solution est la manière de faire correspondre le motif.

Le premier exemple de requête n'avait qu'une seule solution. On change le motif pour qu'il corresponde à la deuxième requête ([Lien 65](#)) :

```
SELECT ?x ?fname
WHERE {?x <http://www.w3.org/2001/vcard-rdf/3.0#FN> ?fname}
```

Il y a ici quatre solutions, une pour chaque triplet qui possède une propriété VCARD name dans les données source :

```
-----
|x                               |name                               |
=====
|<http://somewhere/RebeccaSmith/>|"Becky Smith" |
|<http://somewhere/SarahJones/>  |"Sarah Jones" |
|<http://somewhere/JohnSmith/>   |"John Smith"  |
|<http://somewhere/MattJones/>   |"Matt Jones"  |
-----
```

Jusqu'ici, avec des motifs de triplet et des motifs de base, chaque variable est définie dans chaque solution. Les solutions d'une requête peuvent être extraites d'une table, mais, dans le cas général, c'est une table où seules quelques lignes possèdent une valeur pour chaque colonne. Toutes les solutions à une requête SPARQL donnée ne doivent pas posséder des valeurs pour chaque variable

dans chaque solution, comme on le verra plus tard.

4.2. Motifs de base

Un motif de base est un ensemble de motifs de triplet. Il correspond à des éléments du graphe si tous les motifs du triplet correspondent à la valeur utilisée à chaque fois que la variable de même nom est utilisée.

```
SELECT ?givenName
WHERE
{
  ?y <http://www.w3.org/2001/vcard-rdf/3.0#Family> "Smith" .
  ?y <http://www.w3.org/2001/vcard-rdf/3.0#Given>
  ?givenName .
}
```

Cette requête ([Lien 66](#)) utilise deux motifs de triplet, chacun étant fini par un point (.) (celui après le dernier peut cependant être omis, comme dans l'exemple précédent). La variable y doit être la même pour chaque correspondance pour ce motif.

Les solutions sont :

```
-----
| givenName |
=====
| "John"    |
| "Rebecca" |
-----
```

4.2.1. Qnames

Il y a un mécanisme raccourci pour écrire de longues URI avec des préfixes. La requête précédente est écrite de manière plus claire comme ceci ([Lien 67](#)) :

```
PREFIX vcard: <http://www.w3.org/2001/vcard-rdf/3.0#>SELECT ?givenName
WHERE
{
  ?y vcard:Family "Smith" .
  ?y vcard:Given ?givenName .
}
```

Il y a un mécanisme de préfixation - les deux parties de l'URI (la déclaration de préfixe et la partie après le : dans le qname) sont concaténées. Ce n'est pas exactement la même chose qu'un qname XML mais utilise la règle du RDF pour convertir un qname en URI par concaténation.

4.2.2. Nœuds anonymes

On change la requête juste pour aussi retourner y, ce qui donne ([Lien 68](#)) :

```
PREFIX vcard: <http://www.w3.org/2001/vcard-rdf/3.0#>SELECT ?y ?givenName
WHERE
{
  ?y vcard:Family "Smith" .
  ?y vcard:Given ?givenName .
}
```


Et des nœuds anonymes apparaissent :

```
-----  
| y | givenName |  
-----  
| _:b0 | "John" |  
| _:b1 | "Rebecca" |  
-----
```

Ils apparaissent comme des qnames un peu spéciaux, préfixés par `_:`. Ce n'est pas le label interne pour le nœud anonyme, c'est l'affichage ARQ qui a assigné les `_:b0` et `_:b1` pour montrer quand deux nœuds anonymes sont identiques. Ici, ils sont différents. Cela ne révèle pas le label interne utilisé pour ces nœuds anonymes, bien qu'il soit disponible dans l'API Java.

5. Contraintes sur les valeurs

Aussi appelées filtres.

Cette section décrit comment les valeurs dans une solution peuvent être restreintes. Il y a de nombreuses comparaisons possibles, on se limitera à deux cas ici.

5.1. Correspondance de chaînes

SPARQL fournit une opération pour tester les chaînes, basée sur les expressions régulières. Cela inclut la possibilité de tests comme la commande SQL LIKE, même si la syntaxe de l'expression régulière est différente du SQL :

```
FILTER regex(?x, "pattern" [, "flags"])
```

L'argument `flags` est optionnel. Le drapeau `i` signifie qu'une correspondance sans tenir compte de la casse sera effectuée.

La requête d'exemple trouvera les noms avec un `r` ou un `R` : [Lien 69](#).

```
PREFIX vcard: <http://www.w3.org/2001/vcard-  
rdf/3.0#>SELECT ?g  
WHERE  
{ ?y vcard:Given ?g .  
  FILTER regex(?g, "r", "i") }
```

Avec les résultats :

```
-----  
| g |  
-----  
| "Rebecca" |  
| "Sarah" |  
-----
```

Le langage d'expressions régulières est le même que celui de Xquery ([Lien 70](#)), qui est une version codifiée de celle de Perl. La correspondance d'ARQ est effectuée par le paquet Jakarta ORO : [Lien 71](#).

5.2. Test de valeurs

Des fois, on veut que l'application filtre sur la valeur d'une variable. Dans le fichier de données `vc-db-2.rdf` ([Lien 72](#)), on a ajouté un champ supplémentaire pour l'âge. Il n'est

pas défini par le schéma VCARD, on a donc créé une nouvelle propriété pour ce tutoriel. RDF autorise de mixer différentes définitions de l'information, parce que les URI sont uniques. Notez aussi que la valeur de la propriété `info:age` est typée. Dans cet extrait de données, on veut montrer la valeur typée. Elle peut être écrite en dur, `23` :

```
<http://somewhere/RebeccaSmith/>  
info:age "23"^^xsd:integer ;  
vCard:FN "Becky Smith" ;  
vCard:N [ vCard:Family "Smith" ;  
          vCard:Given "Rebecca" ] .
```

Ainsi, une requête ([Lien 73](#)) pour trouver les gens âgés d'au moins vingt-quatre ans est :

```
PREFIX info: <http://somewhere/peopleInfo#>SELECT  
?resource  
WHERE  
{  
  ?resource info:age ?age .  
  FILTER (?age >= 24)  
}
```

L'expression arithmétique doit être entre parenthèses. La seule solution est :

```
-----  
| resource |  
-----  
| <http://somewhere/JohnSmith/> |  
-----
```

Une seule correspondance, ce qui résulte en l'URI de John Smith. En transformant cette requête pour les gens âgés de moins de vingt-quatre ans ne retourne que Rebecca Smith, rien à propos de Jones.

La base de données ne contient aucune information sur l'âge de Jones : il n'y a pas de propriété `info:age` sur sa VCARD, la variable `age` n'a donc pas reçu de valeur et n'a pas été testée par le filtre.

6. Informations optionnelles

RDF ne représente que des informations semi-structurées, ainsi SPARQL peut trouver des données et ne pas échouer quand elles n'existent pas. La requête utilise une partie optionnelle pour étendre les informations trouvées dans la solution d'une requête mais pour retourner des informations non optionnelles.

6.1. OPTIONAL

Cette requête ([Lien 74](#)) retourne le nom d'une personne, ainsi que son âge, si cette information est disponible.

```
PREFIX info: <http://somewhere/peopleInfo#>  
PREFIX vcard: <http://www.w3.org/2001/vcard-  
rdf/3.0#>SELECT ?name ?age  
WHERE  
{  
  ?person vcard:FN ?name .  
  OPTIONAL { ?person info:age ?age }  
}
```

Deux des quatre personnes dans les données ([Lien 75](#))

possèdent la propriété age. Cependant, parce que le motif de triplet pour l'âge est optionnel, il y a une solution pour les gens qui n'ont pas cette information.

```

-----
| name          | age |
=====
| "Becky Smith" | 23  |
| "Sarah Jones" |     |
| "John Smith"  | 25  |
| "Matt Jones"  |     |
-----

```

Si la clause générale n'est pas ici, aucune information sur l'âge n'aurait été récupérée. Si le motif de triplet avait été inclus sans être optionnel, on aurait eu cette requête ([Lien 76](#)) :

```

PREFIX info: <http://somewhere/peopleInfo#>
PREFIX vcard: <http://www.w3.org/2001/vcard-rdf/3.0#>
SELECT ?name ?age
WHERE
{
  ?person vcard:FN ?name .
  ?person info:age ?age .
}

```

Avec ces deux seules solutions :

```

-----
| name          | age |
=====
| "Becky Smith" | 23  |
| "John Smith"  | 25  |
-----

```

En effet, la propriété info:age doit être présente dans toute solution.

6.2. OPTIONAL et FILTER

OPTIONAL est un opérateur binaire qui combine deux motifs de graphe. Le motif optionnel est un motif de groupe quelconque et peut faire usage d'autres types de motifs SPARQL. Si le groupe correspond, la solution est étendue ; sinon, la solution d'origine est donnée. La requête ([Lien 77](#)) :

```

PREFIX info: <http://somewhere/peopleInfo#>
PREFIX vcard: <http://www.w3.org/2001/vcard-rdf/3.0#>
SELECT ?name ?age
WHERE
{
  ?person vcard:FN ?name .
  OPTIONAL { ?person info:age ?age . FILTER ( ?age > 24 ) }
}

```

Ainsi, tant qu'on filtre sur les âges supérieurs à vingt-quatre ans dans la partie optionnelle, on va toujours obtenir quatre solutions (du motif vcard:FN), mais en ne récupérant l'âge que s'il passe le test :

```

-----
| name          | age |
=====
| "Becky Smith" |     |
-----

```

```

| "Sarah Jones" |     |
| "John Smith"  | 25  |
| "Matt Jones"  |     |
-----

```

Pas d'âge pour Becky Smith, vu qu'il est inférieur à vingt-quatre.

Si la condition de filtre est déplacée hors de la partie optionnelle, alors elle peut influencer le nombre de solutions, mais il peut être nécessaire de rendre le filtre plus compliqué pour que la variable ne soit pas rencontrée, requête ([Lien 78](#)) :

```

PREFIX info: <http://somewhere/peopleInfo#>
PREFIX vcard: <http://www.w3.org/2001/vcard-rdf/3.0#>
SELECT ?name ?age
WHERE
{
  ?person vcard:FN ?name .
  OPTIONAL { ?person info:age ?age . }
  FILTER ( !bound(?age) || ?age > 24 )
}

```

Si la solution possède une variable age, elle doit être supérieure à vingt-quatre. Elle peut aussi ne pas être rencontrée. Il y a maintenant trois solutions :

```

-----
| name          | age |
=====
| "Sarah Jones" |     |
| "John Smith"  | 25  |
| "Matt Jones"  |     |
-----

```

Évaluer une expression qui possède des variables non rencontrées sans s'y attendre cause une exception d'évaluation et toute l'expression échoue.

6.3. OPTIONAL et requêtes dépendant de l'ordre

Il faut faire attention à une chose : l'utilisation de la même variable dans deux clauses optionnelles ou plus.

```

PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX vCard: <http://www.w3.org/2001/vcard-rdf/3.0#>
SELECT ?name
WHERE
{
  ?x a foaf:Person .
  OPTIONAL { ?x foaf:name ?name }
  OPTIONAL { ?x vCard:FN ?name }
}

```

Si le premier OPTIONAL trouve une valeur pour ?name et ?x, le second est une tentative de faire correspondre les mêmes triplets (?x et ?name possèdent des valeurs). Si le premier OPTIONAL n'a pas fait correspondre la partie optionnelle, alors la seconde est une tentative de faire correspondre son triplet avec deux variables.

7. Alternatives dans un motif

Une autre manière de gérer les données semi-structurées est d'effectuer une requête pour une des possibilités. Cette section couvre le motif UNION, où l'une des possibilités

est triée. Tant le vocabulaire vCard que le vocabulaire FOAF ont des propriétés pour le nom des personnes. Dans vCard, il y a vCard: FN, le « nom formaté », et dans FOAF, il y a foaf:name. Dans cette section, nous nous pencherons sur un petit ensemble de données où les noms des personnes peuvent être donnés soit par le vocabulaire FOAF soit par le vocabulaire vCard.

Supposons que nous ayons un graphe RDF ([Lien 79](#)) qui contient des informations en utilisant le nom à la fois avec le vocabulaire vCard et le vocabulaire FOAF.

```
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix vcard: <http://www.w3.org/2001/vcard-rdf/3.0#> .
_:a foaf:name "Matt Jones" .
_:b foaf:name "Sarah Jones" .
_:c vcard:FN "Becky Smith" .
_:d vcard:FN "John Smith" .
```

Une requête pour accéder aux informations du nom, quand il peut être dans les deux formes, pourrait être (q-union1.rq : [Lien 80](#)) :

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX vcard: <http://www.w3.org/2001/vcard-rdf/3.0#>
SELECT ?name
WHERE
{
  { [] foaf:name ?name } UNION { [] vcard:FN ?name }
}
```

Cela renvoie les résultats :

```
-----
| name          |
=====
| "Matt Jones" |
| "Sarah Jones" |
| "Becky Smith" |
| "John Smith" |
-----
```

La forme utilisée pour le nom n'avait pas d'importance, la variable ?name est un ensemble. Ceci peut être réalisé en utilisant un FILTER comme cette requête (q-union1alt.rq : [Lien 81](#)) le montre :

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX vcard: <http://www.w3.org/2001/vcard-rdf/3.0#>
SELECT ?name
WHERE
{
  [] ?p ?name
  FILTER ( ?p = foaf:name || ?p = vcard:FN )
}
```

On teste si la propriété est une URI ou autre chose. Les solutions ne peuvent pas sortir dans le même ordre. La première forme est susceptible d'être plus rapide, en fonction des données et du stockage utilisé, parce que la seconde forme peut obtenir tous les triplets du graphe qui correspondent au modèle des triplets avec des variables non liées (ou nœuds anonymes) dans chaque emplacement, puis de tester chaque ?p pour voir s'il correspond à l'une des valeurs. Il dépendra de la sophistication de l'optimiseur de requêtes de savoir si les requêtes qu'il peut effectuer de manière plus efficace sont capables de passer par la

contrainte ainsi que par la couche de stockage.

7.1. Union - se rappeler où les données ont été trouvées

L'exemple ci-dessous utilise la même variable dans chaque branche. Si les différentes variables sont utilisées, l'application peut découvrir quel sous-modèle cause la correspondance (q-union2.rq : [Lien 82](#)) :

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX vcard: <http://www.w3.org/2001/vcard-rdf/3.0#>
SELECT ?name1 ?name2
WHERE
{
  { [] foaf:name ?name1 } UNION { [] vcard:FN ?name2 }
}
```

```
-----
| name1         | name2         |
=====
| "Matt Jones"  |               |
| "Sarah Jones" |               |
|               | "Becky Smith" |
|               | "John Smith"  |
-----
```

Cette deuxième requête a conservé des renseignements où le nom de la personne vient de l'attribution du nom pour différentes variables.

7.2. OPTIONAL et UNION

Dans la pratique, OPTIONAL est plus commun que UNION, mais ils ont tous deux leur utilisation. OPTIONAL est utile pour augmenter les solutions trouvées, UNION est utile pour la concaténation des solutions à partir de deux possibilités. Ils ne retournent pas les informations nécessaires de la même manière (requête q-union3.rq : [Lien 83](#)) :

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX vcard: <http://www.w3.org/2001/vcard-rdf/3.0#>
SELECT ?name1 ?name2
WHERE
{
  ?x a foaf:Person
  OPTIONAL { ?x foaf:name ?name1 }
  OPTIONAL { ?x vcard:FN ?name2 }
}
```

```
-----
| name1         | name2         |
=====
| "Matt Jones"  |               |
| "Sarah Jones" |               |
|               | "Becky Smith" |
|               | "John Smith"  |
-----
```

Mais méfiez-vous d'utiliser ?name dans chaque OPTIONAL parce que c'est une requête dépendant de l'ordre.

8. Graphes nommés

Cette section couvre les ensembles de données RDF - un ensemble de données RDF est l'unité qui est interrogée par

une requête SPARQL. Il s'agit d'un graphique par défaut et d'un certain nombre de graphes nommés.

8.1. Interrogation des ensembles de données

L'opération de correspondance des graphes (modèles de base ([Lien 84](#)), OPTIONAL ([Lien 85](#)) et UNION ([Lien 86](#))) fonctionne sur un graphe RDF. Cela commence comme le graphe par défaut de l'ensemble de données, mais il peut être modifié par le mot-clé GRAPH.

```
GRAPH uri { ... pattern ... }
```

```
GRAPH var { ... pattern ... }
```

Si une URI est donnée, le modèle sera en correspondance avec le graphe de l'ensemble des données qui portent ce nom - s'il n'y en a pas, la clause GRAPH ne correspond à rien.

Si une variable est donnée, tous les graphes nommés (et non le graphe par défaut) sont jugés. La variable peut être utilisée ailleurs de sorte que si, lors de l'exécution, sa valeur est déjà connue pour une solution, seul le graphe nommé spécifique est essayé.

8.1.1. Exemple de données

Un ensemble de données RDF peut prendre diverses formes. Deux configurations communes : le graphe par défaut est l'union (la fusion RDF) de tous les graphes nommés ou le graphe par défaut est un inventaire des graphes nommés (d'où ils venaient, où ils ont été lus, etc.) Il n'y a pas de limites - un graphe peut être inclus deux fois sous des noms différents, des graphes peuvent aussi partager d'autres triplets.

Dans les exemples ci-dessous nous allons utiliser les données suivantes qui pourraient se produire pour les détails d'un agrégateur RDF de livres.

Grappe par défaut (ds-dft.ttl : [Lien 87](#)) :

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix xsd:
<http://www.w3.org/2001/XMLSchema#> .
<ds-ng-1.ttl> dc:date "2005-07-14T03:18:56+0100"^^xsd:dateTime .
<ds-ng-2.ttl> dc:date "2005-09-22T05:53:05+0100"^^xsd:dateTime .
```

Grappe nommée (ds-ng-1.ttl : [Lien 88](#)) :

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
[] dc:title "Harry Potter and the Philosopher's Stone" .
[] dc:title "Harry Potter and the Chamber of Secrets" .
```

Grappe nommée (ds-ng-2.ttl : [Lien 89](#)) :

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
[] dc:title "Harry Potter and the Sorcerer's Stone" .
[] dc:title "Harry Potter and the Chamber of Secrets" .
```

Maintenant, on a deux petits graphes décrivant des livres

et un graphe par défaut qui enregistre la dernière lecture de ces graphes.

Les requêtes peuvent être exécutées avec l'application en ligne de commande (ce qui se ferait en une seule ligne) :

```
java -cp ... arq.sparql
--graph ds-dft.ttl --namedgraph ds-ng-1.ttl
--namedgraph ds-ng-2.ttl
--query query file
```

Les jeux de données ne sont pas obligatoirement créés juste pour la durée de vie de la requête. Ils peuvent être créés et stockés dans une base de données, ce qui serait plus habituel pour un agrégateur.

8.1.2. Accès à l'ensemble de données

Le premier exemple accède simplement au graphe par défaut (q-ds-1.rq : [Lien 90](#)) :

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX : <.>
SELECT *
{ ?s ?p ?o }
```

Le « PREFIX : <.> » permet simplement de formater la sortie.

s	p	o
:ds-ng-2.ttl	dc:date	"2005-09-22T05:53:05+01:00"^^xsd:dateTime
:ds-ng-1.ttl	dc:date	"2005-07-14T03:18:56+01:00"^^xsd:dateTime

Voici le graphe par défaut seulement - rien depuis les graphes nommés, parce qu'ils ne sont pas interrogés sauf indication contraire par GRAPH.

On peut effectuer une requête pour les triplets en interrogeant le graphe par défaut et les graphes nommés (q-ds-2.rq : [Lien 91](#)) :

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX : <.>
SELECT *
{
  { ?s ?p ?o } UNION { GRAPH ?g { ?s ?p ?o } }
}
```

Donne :

s	p	o	g
:ds-ng-2.ttl	dc:date	"2005-09-22T05:53:05+01:00"^^xsd:dateTime	
:ds-ng-1.ttl	dc:date	"2005-07-14T03:18:56+01:00"^^xsd:dateTime	
_:b0	dc:title	"Harry Potter and the Sorcerer's Stone"	:ds-ng-2.ttl
_:b0	dc:title	"Harry Potter and the Chamber of Secrets"	:ds-ng-2.ttl

			and the	
			Sorcerer's	
			Stone"	
_:b1	dc:title	"Harry Potter	:ds-ng-2.ttl	
			and the	
			Chamber of	
			Secrets"	
_:b2	dc:title	"Harry Potter	:ds-ng-1.ttl	
			and the	
			Chamber of	
			Secrets"	
_:b3	dc:title	"Harry Potter	:ds-ng-1.ttl	
			and the	
			Philosopher's	
			Stone"	

8.1.3. Interrogation d'un graphe spécifique

Si l'application connaît le nom de graphe, elle peut demander directement une requête comme trouver tous les titres dans un graphe donné (q-ds-3.rq : [Lien 92](#)) :

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX : <.>SELECT ?title
{
  GRAPH :ds-ng-2.ttl
  { ?b dc:title ?title }
}
```

Résultats :

title
"Harry Potter and the Sorcerer's Stone"
"Harry Potter and the Chamber of Secrets"

8.1.4. Interrogation des données à partir de graphes qui correspondent à un modèle

Le nom des graphes qui peuvent être interrogés peut être déterminé avec la requête elle-même. Le même processus s'applique pour les variables si elles font partie d'un motif du graphe ou d'un GRAPH. La requête ci-dessous ([Lien 93](#)) fixe une condition sur la variable utilisée pour sélectionner des graphes nommés sur base des informations contenues dans le graphe par défaut.

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX : <.>SELECT ?date ?title
{
  ?g dc:date ?date . FILTER (?date > "2005-08-01T00:00:00Z"^^xsd:dateTime )
  GRAPH ?g
  { ?b dc:title ?title }
}
```

Les résultats de l'exécution de cette requête sur l'ensemble des données de l'exemple sont les titres dans l'un des graphes, le premier avec la date au plus tard le 1 août 2005.

date	title
"2005-09-22T05:53:05 +01:00"^^xsd:dateTime	"Harry Potter and the Sorcerer's Stone"
"2005-09-22T05:53:05 +01:00"^^xsd:dateTime	"Harry Potter and the Chamber of Secrets"

8.2. Décrire les ensembles de données RDF - FROM et FROM NAMED

Une exécution de la requête ne peut donner l'ensemble des données lorsque l'objet de l'exécution est construit ou qu'il peut être décrit dans la requête elle-même. Lorsque les détails sont sur la ligne de commande, un ensemble de données temporaire est créé, mais une application peut créer des ensembles de données et les utiliser ensuite dans de nombreuses requêtes.

Lorsqu'il est décrit dans la requête, FROM url est utilisé pour identifier le contenu dans le graphe par défaut. Il peut y avoir plus d'une clause FROM et le graphe par défaut est la suite de la lecture de chaque fichier dans le graphe par défaut. Il s'agit de la fusion RDF des graphes individuels.

Ne soyez pas troublés par le fait que le graphe par défaut soit décrit par une ou plusieurs URL dans les clauses FROM. C'est à partir de là que les données sont lues, pas le nom du graphe. Comme plusieurs clauses FROM peuvent être données, les données peuvent être lues à partir de plusieurs endroits, mais aucun d'entre eux ne deviendra le nom du graphe.

FROM NAMED url est utilisé pour identifier un graphe nommé. Le graphe est donné par le nom url et les données sont lues à partir de cet emplacement. Des clauses multiples FROM NAMED occasionnent des graphes multiples ajoutés à l'ensemble de données.

Notez que les graphes sont chargés avec le FileManager de Jena, qui peut fournir des sites alternatifs pour les fichiers. Par exemple, la requête peut préciser FROM NAMED <http://example/data> et avoir ses données lues à partir de file:local.rdf. Le nom du graphe sera <http://example/data>, comme dans la requête.

Par exemple, la requête pour trouver tous les triplets dans les deux graphes par défaut et dans les graphes nommés peut être écrite comme ceci ([Lien 94](#)) :

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX : <.>SELECT *
FROM <ds-dft.ttl>
FROM NAMED <ds-ng-1.ttl>
FROM NAMED <ds-ng-2.ttl>
{
  { ?s ?p ?o } UNION { GRAPH ?g { ?s ?p ?o } }
}
```

9. Résultats

SPARQL possède quatre formes de résultat :

- SELECT : retourne un tableau de résultats ;
- CONSTRUCT : retourne un graphe RDF, basé sur

- un modèle dans la requête ;
- DESCRIBE : retourne un graphe RDF, basé sur ce que le processeur de requêtes est configuré pour renvoyer ;
- ASK : pose une requête booléenne.

La forme SELECT retourne directement une table de solutions comme un jeu de résultats, tandis que DESCRIBE et CONSTRUCT utilisent les résultats de l'appariement pour construire des graphes RDF.

9.1. Modificateurs de solutions

La correspondance de modèles produit un ensemble de solutions. Cet ensemble peut être modifié de diverses manières :

- Projection : ne garder que les variables sélectionnées ;
- OFFSET / LIMIT : couper le nombre de solutions (à utiliser avec ORDER BY) ;
- ORDER BY : trier les résultats ;
- DISTINCT : ne rendre qu'une seule ligne pour une combinaison de variables et de valeurs.

Les modificateurs de solution OFFSET / LIMIT et ORDER BY s'appliquent toujours à toutes les formes de résultats.

9.1.1. OFFSET et LIMIT

Un ensemble de solutions peut être abrégé en spécifiant un OFFSET (index de début) et une LIMIT (le nombre de solutions à retourner). L'utilisation de LIMIT seul peut être utile pour assurer que de trop nombreuses solutions ne soient retournées et, ainsi, restreindre l'effet d'une situation inattendue. LIMIT et OFFSET peuvent être utilisés en addition avec le tri pour prendre un intervalle défini par les solutions trouvées.

9.1.2. ORDER BY

Les solutions SPARQL sont triées par l'expression, y compris les fonctions personnalisées.

```
ORDER BY ?x ?y
```

```
ORDER BY DESC(?x)
```

```
ORDER BY x:func(?x) # condition de tri
personnalisé
```

9.1.3. DISTINCT

Le formulaire de résultat SELECT peut prendre le modificateur DISTINCT, qui assure qu'il n'y a pas deux solutions retournées qui sont les mêmes - cela a lieu après la projection pour les variables demandées.

9.2. SELECT

La forme du résultat de SELECT est une projection, avec DISTINCT appliqué, de l'ensemble solution. SELECT identifie les variables nommées dans le jeu de résultats. « * » signifie toutes les variables nommées (les nœuds anonymes dans la requête sont utilisés comme variables d'appariement, mais ne sont jamais retournés).

9.3. CONSTRUCT

CONSTRUCT construit un graphe RDF basé sur un modèle de graphe. Le modèle de graphe peut avoir des variables qui sont liées par une clause WHERE. L'effet est de calculer le fragment de graphe, étant donné le modèle, pour chaque solution de la clause WHERE, après la prise en compte des modificateurs de solution. Les fragments de graphe, dont un pour la solution, sont fusionnés en un seul graphe RDF, le résultat.

Tous les nœuds anonymes explicitement mentionnés dans le modèle de graphe sont créés à nouveau chaque fois que le modèle est utilisé pour une solution.

9.4. DESCRIBE

La forme CONSTRUCT prend un modèle d'application pour les résultats sous forme de graphe. La forme DESCRIBE crée également un graphe, mais sa forme est fournie par le processeur de requêtes, pas par l'application. Pour chaque URI trouvée ou explicitement mentionnée dans la clause DESCRIBE, le processeur de requête devrait fournir un fragment utile de RDF, comme tous les détails connus d'un livre. Le gestionnaire ARQ permet l'écriture d'une description spécifique du domaine.

9.5. ASK

Le formulaire de résultat ASK retourne un booléen, vrai pour un motif qui correspond, faux sinon.

Retrouvez la traduction de Thibaut Cuvelier et Julien Plu en ligne : [Lien 95](#)

Débuter avec les Enlightenment Foundation Libraries (EFL)

Les EFL (Enlightenment Foundation Libraries) sont des bibliothèques graphiques C (des bindings sont également disponibles pour C++, Python, Perl, JavaScript, Vala et Ruby) développées à côté du projet Enlightenment lui-même. Comparables à Qt ou GTK, elles ont été créées pour réaliser des IU (Interface Utilisateur) fluides, du fait de leur base asynchrone. Ce tutoriel montre par étapes comment prendre en main aisément et rapidement ces bibliothèques.

1. Introduction

1.1. Enlightenment et les EFL, qu'est-ce ?

Enlightenment, également connu sous le nom de E (actuellement E17), est un gestionnaire de fenêtrage pour Linux/X11 comprenant une suite complète de bibliothèques pour la réalisation d'interfaces utilisateur aussi fluides qu'esthétiques. Gestionnaire de fenêtrage rapide et léger en publication roulante (*rolling release*) depuis dix ans, E permet notamment d'effectuer un « thémage » complet.

Les EFL (*Enlightenment Foundation Libraries*) sont des bibliothèques graphiques C (des *bindings* sont également disponibles pour C++ ([Lien 96](#)), Python ([Lien 97](#)), Perl, JavaScript et Ruby) développées à côté du projet Enlightenment lui-même. Elles ont été créées pour réaliser des IU (Interface Utilisateur) fluides, du fait de leur base asynchrone et peuvent être utilisées avec des systèmes de bureau tels que GNOME ou KDE. Jugeant que les plateformes fixes commencent à perdre de leur prestige, cédant peu à peu leur place aux plateformes mobiles, les EFL s'orientent vers un support de ces dernières, ayant pour objectif une meilleure portabilité des sources. Ainsi, elles s'orientent principalement vers une utilisation tactile, sans pour autant délaisser l'utilisation bureautique. Elles restent parfaitement adaptées au développement bureautique. Les EFL sont sponsorisées par un certain nombre d'entreprises ([Lien 98](#)), dont Samsung et Free. Pour finir, elles constituent le sujet de ce cours ; nous ne nous intéresserons pas au gestionnaire de fenêtrage lui-même.



Exemple « Buttons » d'Elementary_test

Note : les produits d'Enlightenment ne sont pas

uniquement conçus pour les plateformes mobiles et pour Linux/X11. En effet, ils sont également portés sous Windows et sous Mac OS.

1.2. Les bibliothèques composants les EFL

Comme leur nom l'indique, les EFL sont constituées de plusieurs bibliothèques ayant chacune leurs spécificités :

- Evas ;
- Eina ;
- Edje ;
- Eet ;
- Ecore ;
- Efreet ;
- E_Dbus ;
- Embryo ;
- Eeze ;
- Elementary ;
- et bien d'autres.

À la suite de ce tutoriel seront brièvement présentées certaines de ces bibliothèques.

1.2.1. Eina

La bibliothèque Eina correspond à la trousse à outils des EFL, permettant des opérations core, soit la manipulation de types de données tels que les strings buffers, les stringshares (optimisation sur les chaînes de caractères par un moyen de stockage et d'utilisation à travers toute l'application, réduisant considérablement les doublons de la chaîne en mémoire), listes, itérateurs, fichiers, etc.

Consulter la documentation d'Eina : [Lien 99](#).

1.2.2. Eet

La bibliothèque Eet est un moyen simple et rapide de stockage et de chargement de tout type de données. C'est notamment un moyen de sérialisation. Un exemple d'utilisation peut être trouvé sur le SVN : [Lien 100](#).

Consulter la documentation d'Eet : [Lien 101](#).

1.2.3. Evas

La bibliothèque Evas est une bibliothèque de gestion d'objets graphiques simples (rectangles, images, textes, polygones et lignes) dont la particularité est d'optimiser le rendu en n'en effectuant un que lorsque des objets ont été

modifiés, par souci de consommation de ressources. On n'a donc jamais de rendu partiel, ce qui correspond à un gain de temps et d'énergie important.

Attention à ne pas confondre cette bibliothèque avec la bibliothèque Elementary : Evas n'est pas conçue pour la création de widgets mais pour la création et gestion d'éléments simples dont la liste exhaustive a été donnée plus haut.

À noter qu'Evas est la base de tous les éléments graphiques fournis par les EFL.

Consulter la documentation d'Evas : [Lien 102](#).

1.2.4. Ecore

La bibliothèque Ecore est une bibliothèque de boucles d'évènements permettant l'utilisation de timers, d'animations, de fonctionnalités réseaux (tcp, udp, http, ftp, etc.) et de bien d'autres choses. L'intérêt fondamental que cette bibliothèque fournit est que rien de ce qu'elle met à disposition n'est bloquant, ce qui simplifie la réalisation d'applications fluides. Par exemple, quand Ecore n'a plus aucune tâche à effectuer, il rentre en mode Idle, ce qui évite une consommation inutile de ressources.

Consulter la documentation d'Ecore : [Lien 103](#).

1.2.5. Edje

La bibliothèque Edje est sans doute la bibliothèque la plus importante des EFL, dans le sens où toute application utilisant les EFL va y recourir à un moment ou à un autre. Elle correspond à un moyen de concevoir des applications en séparant la partie design de la partie concernant la logique et le code. En effet, la création d'une interface graphique avec Edje ne nécessite pas la moindre ligne de code C, ce qui permet notamment aux développeurs et aux designers de travailler ensemble sur un projet. Le designer mettrait en place la partie Edje de l'application tandis que le développeur pourrait se concentrer exclusivement sur le code C, ce qui correspondrait à un gain non négligeable de temps. Quel développeur n'a jamais eu de problème avec son chef d'équipe comme quoi le design de l'application ne collerait pas avec le travail initial des designers ? Grâce à la bibliothèque Edje, c'en est fini de ce type de problèmes.

Le codage avec la bibliothèque Edje se fait un peu comme en CSS avec une association *propriété: valeur*. Exemple du traditionnel *Hello world* :

```
collections {
  group {
    name: "interface";
    parts {
      part {
        name: "text";
        type: TEXT;
        description {
          state: "default" 0.0;
          color: 255 255 255 255;
          text {
            font: "Sans-serif";
            text: "Hello world !";
            size: 18;
          }
        }
      }
    }
  }
}
```

```
}
}
}
}
```

Et le rendu :



La bibliothèque Edje utilise plusieurs bibliothèques fournies par les EFL :

- Eet pour la partie concernant le stockage de données ;
- Evas pour la partie graphique des thèmes écrits avec Edje ;
- Ecore pour gérer la boucle d'évènements.

Cette bibliothèque sera bien plus amplement détaillée par la suite ; cette partie consiste uniquement en un aperçu de la chose.

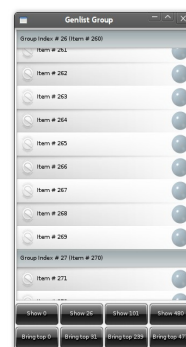
Consulter la documentation d'Edje : [Lien 104](#).

1.2.6. Elementary

La bibliothèque Elementary met à disposition un grand nombre de widgets hautement personnalisables. Certains widgets d'Elementary, nommés « widgets performants », ont initialement été conçus pour être utilisés sur des plateformes mobiles, nécessitant une grande fluidité.

Pour illustrer, on peut prendre un exemple qui a poussé les développeurs à mettre en place les *genlists* : on dispose d'une liste de plusieurs centaines d'éléments graphiques. Parmi ces éléments, on souhaite uniquement en afficher une vingtaine en même temps à l'écran. Dans ce cas de figure, n'est-ce pas être inconscient que de vouloir prendre de la mémoire pour un tel nombre ? Le principe est de stocker les items dans un arbre pour éviter d'avoir à parcourir trop d'éléments. Les widgets performants vont être créés lors de l'affichage, et détruits lorsqu'ils ne seront plus affichés (à noter que derrière cela, on a tout de même un principe de cache).

Voici une capture d'écran de l'exemple « Genlist Group » d'Elementary_test qui permet de visualiser une genlist ordinaire :



Consulter la documentation d'Elementary : [Lien 105](#).

1.3. Différences avec des frameworks tels que Qt et GTK

Il est fort possible que vous vous interrogiez sur la différence qu'ont les EFL avec les gros frameworks d'aujourd'hui tels que Qt et GTK. Certes, ces trois frameworks permettent plus ou moins d'effectuer la même chose, mais EFL ne tente pas de suivre ces deux monstres. Il préfère se démarquer d'eux sur plusieurs points détaillés ci-dessous de manière non exhaustive.

Comme Qt et GTK, les EFL fournissent avec leur bibliothèque Elementary des widgets (comme des boutons, cases à cocher, etc.). Il faut voir les widgets des EFL comme des outils et non comme de simples éléments de fenêtre, à la différence de ce que proposent Qt et GTK. Avec Qt, on voit couramment des widgets conteneurs présents dans d'autres conteneurs (exemple : un QPushButton présent dans un QWidget, lui-même englobé dans un QMainWindow), soit des arbres de widgets s'englobant les uns les autres. Avec EFL, les widgets sont utilisés comme des « briques » combinées avec la bibliothèque Edje. Les widgets d'Elementary sont hautement personnalisables, donnant la possibilité de modifier l'échelle, de changer diverses propriétés. Une autre différence notable avec les frameworks concurrents est sans doute que les EFL présentent pour leurs widgets une zone tactile différenciée de la zone d'affichage, peu importe le fait que le tactile soit utilisé ou non dans l'application.

Il est logique de constater une grande similitude entre Qt Quick (de Qt) et du code de thèmes avec Edje (à noter qu'Edje existait avant que Qt Quick ne fasse son entrée - c'est par ignorance que bien des gens ont annoncé avoir attendu le concept utilisé par QML). Il existe toutefois de nombreux points les dissociant. Tout d'abord, les fichiers de description d'Edje sont compilés avant de pouvoir être observés, à la différence des fichiers QML. Avec Edje, il est possible de modifier entièrement le thème d'une application sans avoir besoin de modifier le code C. On peut passer d'une architecture en liste à une architecture tout autre d'une simple modification du fichier Edje. Quant à lui, Qt Quick va plutôt se baser sur un moteur JavaScript. Glade, l'équivalent de Qt Designer, écrit ses fichiers en XML. Sur ce point, Edje et QML sont donc plus avancés dans le sens où un fichier Edje permet de réaliser le thème de l'application en plus de disposer les éléments dans l'interface utilisateur.

Tel que vous avez dû l'avoir compris, les EFL s'associent au mot « optimisation » par les moyens dont sont réalisés leurs composants et par leur base asynchrone. Même si certaines optimisations peuvent vous paraître peu intéressantes, voire même complètement inutiles, les EFL ont prouvé par cela leur fluidité. Généralement, une application développée avec les EFL, sauf problème de conception de son coeur, ne devrait jamais geler. C'est une grande différence par rapport aux frameworks du type de Qt et de GTK.

2. Installation

2.1. Windows

Si vous souhaitez utiliser les EFL sous Windows, téléchargez cet installateur ([Lien 106](#)), lancez-le et laissez-vous guider. Vous pouvez également vous référer à cette page pour choisir l'installateur adapté à vos besoins : [Lien 107](#).

2.2. Mac OS

Si vous préférez utiliser les EFL sous Mac OS, vous devez être en possession des programmes Fink et XCode2. L'installation des EFL sous Mac OS nécessite des dépendances, ce qui explique l'intérêt de Fink :

```
fink install xorg
fink install m4
fink install autoconf2.5
fink install automake1.9
fink install libtool14
fink install gettext-tools
fink install pkgconfig
fink install libjpeg
fink install libpng3
fink install dbus-dev
```

Liste en provenance de ce site : [Lien 108](#). Vous pouvez alors télécharger les sources :

```
svn co
http://svn.enlightenment.org/svn/e/trunk/eina
eina-svn
svn co
http://svn.enlightenment.org/svn/e/trunk/eet eet-
svn
svn co
http://svn.enlightenment.org/svn/e/trunk/evas
evas-svn
svn co
http://svn.enlightenment.org/svn/e/trunk/ecore
ecore-svn
svn co
http://svn.enlightenment.org/svn/e/trunk/efreet
efreet-svn
svn co
http://svn.enlightenment.org/svn/e/trunk/embryo
embryo-svn
svn co
http://svn.enlightenment.org/svn/e/trunk/edje
edje-svn
svn co
http://svn.enlightenment.org/svn/e/trunk/e_dbus
e_dbus-svn
svn co
http://svn.enlightenment.org/svn/e/trunk/elementa
ry elementary-svn
```

Cela peut prendre un certain temps. La dernière étape est la compilation et l'installation. Pour cela, il s'agit d'entrer dans chaque répertoire (dans l'ordre donné ci-dessus) et d'exécuter les commandes :

```
./autogen.sh
make
make install && ldconfig
```

Par exemple, pour eina :

```
cd /chemin/vers/eina/  
./autogen.sh  
make  
make install && ldconfig
```

Note : la page mentionnée ci-dessus parle de la nécessité d'un hack dans `e17/libs/eet/src/lib/eet_lib.c` pour les versions de Max OS X précédant la version Leopard (10.5), donc les versions Tiger (10.4) et précédentes. Ce hack correspond à ajouter les deux lignes suivantes aux alentours de la ligne 21.

```
typedef unsigned int uint8_t;  
typedef unsigned char uint32_t;
```

2.3. Linux

Pour les distributions telles qu'Ubuntu, une solution est d'utiliser le script `easy_e17.sh` ([Lien 109](#)) pour récupérer les sources depuis le SVN puis les compiler. La méthode ne marche pas toujours du fait que des changements dans l'organisation du dépôt cassent de temps en temps le script. Quelques modifications sont donc parfois requises.

```
sudo apt-get install xterm make gcc bison flex  
subversion automake autoconf autotools-dev  
autoconf-archive libtool \  
gettext libpam0g-dev libfreetype6-dev libpng12-  
dev zlib1g-dev libjpeg62-dev libtiff4-dev  
libungif4-dev librsvg2-dev \  
libx11-dev libxcursor-dev libxrender-dev  
libxrandr-dev libxfixes-dev libxdamage-dev  
libxcomposite-dev libxss-dev \  
libxp-dev libxext-dev libxinerama-dev libxft-dev  
libxfont-dev libxi-dev libxv-dev libxkbfile-dev  
libxres-dev \  
libxtst-dev libltdl7-dev libglu1-xorg-dev  
libglut3-dev xserver-xephyr libdbus-1-dev cvs  
subversion mercurial \  
liblua5.1-dev libavformat-dev mplayer libxine-dev  
libxml2-dev libcurl4-openssl-dev wget libexif-dev  
libsqlite3-dev \  
libxine1-all-plugins libxine1-ffmpeg libudev-dev  
liblua5.1-dev doxygen
```

Liste initialement en provenance de ce site : [Lien 110](#) ; doxygen est optionnel. Une fois ces paquets installés, on récupère `easy_e17.sh` :

```
wget  
http://omicon.homeip.net/projects/easy_e17/easy_  
e17.sh
```

À l'heure actuelle, Elementary n'est pas sortie en version stable. Ainsi, cette bibliothèque n'est pas installée par défaut. Remédions-y : éditez le fichier `easy_e17.sh` et ajoutez en dernière position « elementary » dans `e17_basic` :

```
e17_basic="eina eet evas ecore efreet eio eeze  
e_dbus embryo edje elementary"
```

Une fois cela fait, on peut lancer le script :

```
chmod +x easy_e17.sh  
./easy_e17.sh -i
```

Cela peut prendre un certain temps. L'installation du module exchange est susceptible de bloquer `easy_e17.sh`. Si c'est le cas chez vous, retirez-le de la liste des modules à installer puis relancez le script. Vous pourrez toujours l'ajouter manuellement par la suite.

Par défaut, le script va récupérer les sources depuis le SVN et installer les EFL dans `/opt/e17`. Les sources seront disponibles dans `$HOME/e17_src` (par exemple, `/home/username/e17_src`).

Une fois l'exécution du script terminée, le script va suggérer l'ajout de plusieurs variables d'environnement. Pour une installation par défaut, les variables suivantes sont nécessaires :

```
export PATH=/opt/e17/bin:$PATH  
export LD_LIBRARY_PATH=/opt/e17/lib:  
$LD_LIBRARY_PATH  
export PKG_CONFIG_PATH=/opt/e17/lib/pkgconfig:  
$PKG_CONFIG_PATH
```

Les utilisateurs d'Archlinux, la commande suivante suffit à faire l'installation :

```
pacman -S e-svn
```

Toutefois, il faudra également installer Elementary qui n'est pas inclus par défaut.

Pour les distributions ne permettant pas ce type d'installation, se référer à cette page du site officiel : [Lien 111](#).

3. Les forces et les problèmes des EFL

3.1. Les forces

Les EFL sont extrêmement pratiques pour le développement d'applications fluides et esthétiques. Le tactile est très bien géré, d'où le moyen de faire tourner des logiciels faits avec les EFL sous de multiples types de plateformes, fixes ou mobiles. Un logiciel, peu importe la plateforme sur laquelle il doit s'exécuter, n'aura généralement jamais de souci de ralentissement, sauf souci de conception. En effet, les solutions d'optimisation mises à disposition sont nombreuses car initialement non négligées.

Comme autre force, on peut noter que le thémage entièrement personnalisable d'une application développée avec les EFL est extrêmement simple et offre de nombreuses possibilités, comme la réalisation d'animations et d'effets sans avoir besoin de toucher à une seule ligne de C, ce qui permet aux développeurs et aux designers de travailler ensemble.

Enfin, les EFL disposent de bien des modules conçus pour la réalisation de tout type d'application, ce qui permet à leurs utilisateurs de ne pas avoir besoin d'utiliser des bibliothèques externes pour combler des manques. Multimédia, réseau, bureautique et bien d'autres choses sont déjà englobés par les EFL, et la liste ne fait que s'agrandir !

3.2. Les faiblesses

D'un point de vue de la communication, Enlightenment souffre de quelques problèmes. La documentation est assez limitée, les moteurs de recherche ne trouvent pas grand-chose sur les EFL (il faut souvent passer par le mot-clé Enlightenment pour avoir des résultats pertinents). Ces problèmes de communication engendrent le fait qu'il y ait un manque de tutoriels et par exemple que l'installation des EFL ne soit pas toujours évidente.

La bibliothèque Elementary, bien que « bien garnie », manque de widgets, et le non tactile n'est pas tellement une priorité dans le développement.

Pour finir, on peut ajouter qu'il arrive que les développeurs réécrivent des éléments afin de les améliorer, mais retirent occasionnellement des possibilités pour les remplacer par d'autres. L'adaptation n'est pas toujours évidente. Quoi qu'il en soit, il faut noter que les changements majeurs prévus pour E17 ont été repoussés à E18. Ce problème d'adaptation ne devrait ainsi pas se poser avant un certain temps.

4. Introduction à Edje

La description faite dans la partie précédente est suffisante pour comprendre le rôle qu'a Edje dans le développement d'une application avec les EFL. Pour résumer, le principe des EFL est de séparer le design et la partie applicative. Grossièrement dit, on réalise son interface avec Edje sans avoir à écrire la moindre ligne de C puis on conçoit le reste avec Elementary et les autres bibliothèques.

4.1. Les fichiers EDC et EDJ

La plupart des tutoriels disponibles sur la Toile entendent par Edje deux choses et ce cours n'échappera pas à la règle :

- les fichiers de description EDC (*Edje Data Collection*) où est écrite la description du thème de l'application ;
- les fichiers EDJ obtenus lors de la compilation des fichiers EDC.

Note : c'est la bibliothèque libedje.so qui relie les fichiers EDJ et le code C.

4.2. Compilation et vision d'un fichier Edje

Dans cette partie, je vais partir du principe que vous avez un fichier `description.edc` dans un répertoire donné contenant une description de thème quelconque. Si vous souhaitez suivre les étapes de la compilation mais que vous n'avez pas de fichier `.edc` en attente de compilation, vous pouvez par exemple utiliser l'exemple *Hello world* donné précédemment :

```
collections {
  group {
    name: "interface";
    parts {
      part {
        name: "text";
        type: TEXT;
        description {
          state: "default" 0.0;
```

```
        color: 255 255 255 255;
        text {
          font: "Sans-serif";
          text: "Hello world !";
          size: 18;
        }
      }
    }
  }
}
```

La compilation d'un fichier Edje est relativement simple. En réalité, cela se limite à une seule commande, qui est la suivante :

```
edje_cc description.edc
```

Avec cela, si aucune erreur n'est détectée, un fichier `description.edj` a dû faire son apparition dans le même répertoire que le fichier `description.edc`. Pour visionner un fichier EDJ, on utilise généralement `edje_player` :

```
edje_player description.edj
```

Cette ligne exécutée, votre interface devrait s'afficher sous vos yeux.

Il est bon de savoir que divers arguments peuvent être passés à `edje_player` et à `edje_cc` pour altérer des choses et d'autres. Pour en connaître la liste, entrez :

```
edje_cc --help
```

Pour `edje_cc`. Et pour `edje_player` :

```
edje_player --help
```

5. Écriture d'un fichier de description Edje

5.1. Généralités sur les fichiers EDC

Les fichiers de description Edje, donc les fichiers EDC, font parfois penser à une structure en C ou bien à du code CSS dans le sens où ils correspondent à des définitions de propriétés. On obtient des arbres de définitions de la sorte :

```
block
{
  property: value;

  block
  {
    ...
  }

  ...
}
```

Étudions notre fichier `Hello_world.edc` contenant le code de l'exemple *Hello world*.

```
collections { ... }
```

Le mot-clé « collections » correspond à la définition d'un

ensemble de groupes définissant le thème de l'application.

```
group {
    name: "group_name";
    ...
}
```

Ce mot-clé-ci correspond quant à lui à la définition d'un groupe, ou objet, composant le thème. Chaque groupe a un nom unique, défini par le mot-clé « name », et peut contenir divers éléments, comme des « parts » ou des scripts (cette notion sera abordée plus loin). Si un groupe du même nom qu'un autre est défini, il va écraser complètement son prédécesseur, d'où l'unicité de la valeur du mot-clé « name ».

```
parts {
    part {
    }
    ...
}
```

Les « parts » sont un ensemble de « part », chacune d'elles représentant les éléments graphiques présents dans un thème, donc un bouton, label, etc.

```
part {
    name: "text";
    type: TEXT;
    ...
}
```

Chaque « part » possède au minimum un nom (*name*) et un type. Ce dernier peut correspondre à l'un des types suivants :

- RECT : les rectangles ;
- TEXT : les textes ;
- IMAGE : les images ;
- SWALLOW : les conteneurs ;
- TEXTBLOCK : les blocs de texte ;
- GROUP : les groupes ;
- BOX : les boîtes ;
- TABLE : les tableaux ;
- EXTERNAL : les éléments externes.

Dans le cas de l'exemple *Hello world*, on a un TEXT. Selon le type, la *description* de chaque *part* va être menée à varier. Quoi qu'il en soit, les données des propriétés du thème ne sont pas condamnées à garder la valeur qu'on leur a assignée dans le fichier EDC. En effet, le code C généralement situé en parallèle - il est rare de trouver des applications réalisées entièrement avec Edje - fournit bien des moyens de changer leur valeur lors de l'exécution.

Pour plus d'informations concernant les propriétés mises à disposition, vous pouvez consulter cette page de la documentation ([Lien 112](#)) qui en donne la liste exhaustive. Nous allons tout de même nous pencher par la suite sur plusieurs de ces éléments afin d'illustrer.

Notez que l'on peut commenter les fichiers EDC avec des commentaires tels qu'on les écrit en C :

```
// Ceci est un commentaire
/* Ceci en est un autre */
```

Pour ne pas se perdre dans son travail, tout développeur a l'habitude de créer de multiples fichiers pour aérer son code. Pour cela, Edje permet l'utilisation de directives telles que la directive `#include`, dont voici un exemple :

```
collections {
    #include "interface.edc"
    ...
}
```

5.2. Points particuliers

Maintenant que nous avons passé en revue une description Edje, nous allons pouvoir entrer dans les détails.

5.2.1. Disposition des éléments

Tout d'abord, un point sur l'affichage à l'écran d'un thème : les éléments sont affichés dans l'ordre de leur écriture dans le fichier Edje. Cela signifie que s'il y a lieu de superposition d'éléments, l'élément défini en dernier sera situé « au-dessus » des autres.

Puisque nous parlons de superposition, parlons de disposition : comment faire pour afficher un élément à une position donnée ? Comment faire pour que notre élément s'agrandisse et se rétrécisse selon la taille de la fenêtre ? Comment disposer un élément relativement à un autre ? Le positionnement et le comportement face aux redimensionnements forment sans aucun doute la partie la plus importante d'une interface utilisateur. Edje nous permet par le biais des propriétés des *descriptions* de la gérer, par de divers moyens.

Dans un premier temps, voyons l'alignement avec les balises « rel1 » et « rel2 ». La balise rel1 correspond au point situé en haut à gauche de l'élément à placer, et rel2 au point en bas à droite. Pour un placement relatif d'un élément à 50 % de l'axe des abscisses et à 0 % de l'axe des ordonnées par rapport à rel1, on va utiliser :

```
rel1 {
    relative: 0.5 0.0;
}
```

Comme vous avez dû le comprendre, 0 % d'un axe va correspondre à la valeur 0 tandis que 100 % va correspondre à 1. Pour un placement absolu à 200x300px de rel1, on va utiliser :

```
rel1 {
    offset: 200 300;
}
```

Qui est l'équivalent logique de :

```
rel1 {
    relative: 0 0;
    offset: 200 300;
}
```

Tout en gardant en tête que les valeurs d'offset peuvent être négatives. Il est possible de combiner les propriétés « relative » et « offset ». Cela correspond tout simplement à un positionnement absolu à partir d'un point relatif.

On peut d'ailleurs aussi faire un placement relatif à un autre élément avec l'aide de la propriété « to » :

```
rel1 {
```

```
...
to: "other part's name";
}
```

Qui est l'équivalent de :

```
rell {
...
to_x: "other part's name";
to_y: "other part's name";
}
```

Les propriétés « `to_x` » et « `to_y` » veulent respectivement dire que l'on souhaite placer notre élément relativement à la position des abscisses (`to_x`) ou bien des ordonnées (`to_y`). Utiliser les deux n'est bien entendu pas obligatoire, on peut utiliser l'un et/ou l'autre, ou tout simplement ne pas les utiliser du tout.

Edje laisse au développeur la possibilité de positionner un élément par rapport à la place qui lui est attribuée avec align. Par défaut, lorsqu'un élément est plus petit que l'espace dont il dispose, il va s'y placer au centre (`align: 0.5 0.5`). Exemple pour placer un élément en haut à gauche de l'espace lui étant alloué :

```
align: 0 0; // 0 pour 0 % et 1 pour 100 %
```

En ce qui concerne les propriétés liées au dimensionnement, on a le choix entre plusieurs éléments. Ci-dessous se trouvent trois des plus importants, en provenance de la documentation ([Lien 113](#)) :

- `fixed [longueur, 0 ou 1] [hauteur, 0 ou 1]` : fixe ou non la longueur et la largeur de l'élément ;
- `min [longueur, px] [hauteur, px]` : fixe une taille minimale à l'élément (astuce : -1 pour ne pas fixer, 1 pour laisser Edje calculer la taille) ;
- `max [longueur, px] [hauteur, px]` : fixe une taille maximale à l'élément.

Pour le comportement face à un redimensionnement, on va plutôt utiliser `aspect`, le ratio entre la longueur et la hauteur, et `aspect_preference`, le comportement de redimensionnement (HORIZONTAL, VERTICAL ou BOTH). Ces deux propriétés sont liées. Exemple d'un rectangle se redimensionnant sur les deux axes tout en conservant ses proportions initiales :

```
collections {
  group {
    name: "interface";
    parts {
      part {
        name: "button";
        type: RECT;
        description {
          state: "default" 0.0;
          aspect: 1.0 1.0;
          aspect_preference: BOTH;
        }
      }
    }
  }
}
```

5.2.2. Les images

La documentation est suffisante pour comprendre comment définir un certain nombre d'éléments. Il peut tout

de même s'avérer intéressant de s'arrêter sur les images afin de parler de certaines fonctionnalités en plus du type lui-même.

```
collections {
  group {
    name: "interface" ;
    images {
      image: "image_normal.png" COMP;
    }
    parts {
      part {
        name: "image";
        type: IMAGE;
        description {
          state: "default" 0.0;
          aspect: 1 1;
          aspect_preference: BOTH;
          image {
            normal: "image_normal.png";
          }
        }
      }
    }
  }
}
```

Cela affiche l'image « `image_normal.png` » s'adaptant à l'écran tout en conservant son ratio d'origine.

```
#define IMAGE_NORMAL "image_normal.png"
collections {
  group {
    name: "interface";
    images {
      image: IMAGE_NORMAL COMP;
    }
    parts {
      part {
        name: "image";
        type: IMAGE;
        description {
          state: "default" 0.0;
          image {
            normal: IMAGE_NORMAL;
          }
        }
      }
    }
  }
}
```

Le code identique avec l'utilisation d'une macro `IMAGE_NORMAL`, dans l'unique but d'informer que les macros existent et s'écrivent tout comme en C.

Afin d'utiliser une image, il est nécessaire de passer au préalable par `images` pour la définir :

```
images {
  image: "image_normal.png" COMP;
}
```

Un certain nombre de formats sont supportés par Edje (JPEG, PNG, etc.), qui permet également une compression de l'objet image. `COMP`, utilisé dans l'exemple, correspond à une compression sans perte, `RAW` à aucune compression, `LOSSY[x]` à une compression de qualité `x` (`x` variant entre 0 et 100), comparable à la qualité JPEG, et `USER` à une image non intégrée au Edje, lue depuis le

disque. Il est souvent préférable d'utiliser COMP plutôt que USER dans le sens où il est plus pratique de ne pas avoir à la charger depuis le disque.

Attention : dans le cas de la qualité RAW, il reste nécessaire de faire attention à la taille finale que peut avoir le fichier binaire.

Puisqu'il est question de compression, et donc de qualité, une question se pose : nous avons une image donnée que l'utilisateur peut redimensionner. Comment faire pour qu'elle ne soit pas floue pour les grands écrans ? Une réponse possible serait de joindre une grande image plutôt qu'une image de qualité moyenne. Dans ce cas, la question inverse se poserait : comment faire pour qu'il n'y ait pas de baisse de la qualité pour les petits écrans ? Edje propose une solution : *set*.

```
collections {
  group {
    name: "interface";
    images {
      set {
        name: "my_image";
        image {
          image: "image-32.png" COMP;
          size: 0 0 32 32;
        }
        image {
          image: "image-64.png" COMP;
          size: 33 33 64 64;
        }
        image {
          image: "image-128.png" COMP;
          size: 65 65 128 128;
        }
      }
    }
  }
  parts {
    part {
      name: "image";
      type: IMAGE;
      description {
        state: "default" 0.0;
        aspect: 1 1;
        aspect_preference: BOTH;
        image {
          normal: "my_image";
        }
      }
    }
  }
}
```

Dans ce code, on définit un *set* d'images de différentes tailles, avec « my_image » comme valeur de la propriété « name ». Au moment de la définition de l'élément IMAGE, on va utiliser la valeur de la propriété « name » de notre *set* :

```
image {
  normal: "my_image";
}
```

De là, Edje va déterminer quelle image choisir selon la taille de la fenêtre à l'exécution à l'aide des propriétés « size » de chaque image du set. Dans le cas où, lors de l'exécution, la taille de la fenêtre ne serait pas incluse dans

l'intervalle des propriétés « size », l'image ayant un min-size à 0 sera utilisée jusqu'à la nécessité d'un changement menant à l'utilisation d'une autre image.

La solution des *set* d'images est très pratique pour éviter d'avoir à passer par une solution bien plus lourde telle que l'utilisation de SVG.

5.2.3. Effets et animations avec Edje

Tout ce qu'on a vu jusque-là est certes assez instructif, mais n'est pas vraiment impressionnant. Cette partie va donc donner l'occasion d'aborder un point que vous attendiez : les animations, les effets, et donc la gestion des événements, tout cela dans un fichier de description Edje sans la moindre ligne de C.

On appelle « programme » une action effectuée par Edje, généralement en réponse à un signal (un clic de la souris, par exemple). Ces *programs* peuvent être intégrés à un *group* et effectuer diverses choses sur divers éléments, comme changer l'état d'un ou plusieurs éléments.

Voyons un exemple simple. On souhaite faire en sorte qu'un élément TEXT affiché à l'écran laisse sa place à un élément IMAGE lorsqu'on clique dessus, et inversement. Ainsi, nous allons devoir définir deux « programmes » : le premier correspondra à la réaction à un clic sur l'élément TEXT et le second à un clic sur l'élément IMAGE.

La méthode n'est pas très compliquée. Chacun des deux éléments, TEXT et IMAGE, va être doté de deux descriptions : une pour l'état par défaut et une autre pour l'état après un clic. Lorsque l'utilisateur va cliquer sur l'élément TEXT, le programme associé à cette interaction va changer l'état des deux éléments à l'état « après un clic » afin de repositionner l'élément TEXT hors de la fenêtre et de repositionner l'élément IMAGE à l'intérieur. Pour un clic sur l'élément IMAGE, l'action est réciproque.

```
collections {
  group {
    name: "interface";
    images {
      image: "image.png" COMP;
    }
    parts {
      part { // Notre élément TEXT
        name: "text";
        type: TEXT;
        mouse_events: 1;
        description { // État par défaut
          state: "default" 0.0;
          text {
            text: "Cliquez ici !";
            size: 12;
            font: "Sans";
          }
        }
        description { // État clicked
          state: "clicked" 0.0;
          inherit: "default" 0.0;
          rell.offset: -1000 0;
        }
      }
      part { // Notre élément IMAGE
        name: "image";
        type: IMAGE;
      }
    }
  }
}
```




Qt SDK 1.1 disponible en version finale

Qt 4.7.3 et Qt Mobility 1.1.3 également

L'équipe de Qt vient d'annoncer la disponibilité de la version finale du kit de développement Qt SDK 1.1

Cette version apporte quelques corrections de bogues à la Release Candidate. Ce SDK est le premier à permettre de créer des applications fondées sur Qt 4.7 et de les publier sur la galerie d'applications Ovi Store, y compris pour les applications Qt Quick.

Qt SDK 1.1 unifie tous les kits de développements Qt actuellement disponibles. De ce fait, il vise à la fois les solutions Desktop et mobiles. Il intègre notamment un paquet complémentaire qui permet l'installation des API natives qui peuvent être utilisées dans un environnement de développement Symbian.

On notera également l'intégration de la compilation à distance, qui permet de compiler un projet pour toutes les plateformes prises en charge par Nokia et ce, que ce soit depuis un environnement de développement Windows, Linux ou Mac.

Cette sortie du SDK Qt version 1.1 est accompagnée de la publication de Qt 4.7.3 et de Qt Mobility 1.1.3 qui sont directement inclus dans Qt SDK 1.1 (Qt Mobility 1.1.3 peut être téléchargé indépendamment).

Qt 4.7.3 et Qt Mobility 1.1.3 apportent uniquement des corrections de bogues importants pour Qt et Qt mobility, en particulier pour la plate-forme mobile Symbian, et tient compte du retour d'expériences des utilisateurs pour la beta et la Release Candidate de Qt SDK 1.1.

Qt Creator 2.2, quant à lui, sera publié très bientôt et nécessitera pas la réinstallation de Qt SDK 1.1 pour une mise à jour.

Qt SDK 1.1 est disponible sur cette page : [Lien 115](#)

Qt Mobility 1.1.3 peut-être téléchargé sur cette page : [Lien 116](#)

Commentez cette news d'Hinault Romaric en ligne : [Lien 117](#)

Sortie de Qwt 6.0.0

La bibliothèque de widgets techniques pour Qt ne supportera plus Qt 3

La version 6.0.0 de Qwt est sortie, une année après la version précédente, la 5.2.1.

La principale nouveauté est l'abandon du support de Qt 3 pour nettoyer l'API. Cette version requiert donc Qt 4.4 ou plus récent. Autre conséquence : le portage d'applications

de Qwt 5 à Qwt 6 ne sera pas aisé.

Qwt 6.0.0 est disponible : [Lien 118](#).

Voici la liste des principaux changements :

- 1) Qt3 support dropped
- 2) QwtPlot layout/render code ported from int to double
Exported/printed documents in scalable formats like SVG or PDF are 100% scalable now.
- 3) Template base classes introduced for curve and curve data to be reusable in all plot items displaying series of samples.
- 4) New plot items
 - QwtPlotHistogram
 - QwtPlotIntervalCurve (error bars or displaying the area between 2 curves)
 - QwtPlotSpectroCurve (mapping the z value to a color)
- 5) Raster items
 - QwtMatrixRasterData introduced
 - More accurate rendering
 - Several API changes
 - Thread support for rendering spectrograms
- 6) QwtPlot::print moved to QwtPlotRenderer
- 7) Other new classes
QwtColumnSymbol
QwtDoublePoint3D
QwtIntervalSymbol
QwtPlotDirectPainter
QwtSamplingThread
QwtSystemClock
- 8) QwtPicker and friends reorganized,
QwtPickerTrackerMachine added for displaying a rubberband for mouse moves. Enter/Leave added to events, that are handled by the picker machines.
- 9) QwtScaleWidget::LayoutFlag added
Introduced to control the direction of vertical axis titles.
- 10) QwtWeedingCurveFitter added
QwtWeedingCurveFitter is an implementation of the Douglas/Peucker algorithm, that can be used to reduce the number of curve points. It can be very useful to improve the performance of painting curves of many lines (e.g. by implementing different level of details).
- 11) Legend code update for representing different pixmapes for different types of plot items.
- 12) Copy operators removed, using pointers instead
- 13) QwtPolarPoint from qwtpolar added
- 14) QwtThermo Optional QwtColorMaps added
- 15) Interfaces and code of all sliders/dials cleaned up.
QApplication::globalStrut(), styled backgrounds ...

Commentez cette news de Thibaut Cuvelier en ligne : [Lien 118](#)

Quelques pensées sur Qt 5, quelles seront ses lignes directrices ?

Six ans déjà que Qt 4.0 est sorti, en juin 2005. Depuis lors, beaucoup de choses ont changé dans l'industrie du logiciel : le développement se déroulait principalement pour le desktop, alors que les périphériques mobiles se sont répandus. L'interface utilisateur est passée de la souris au toucher, du statique au fluide. Sept versions mineures depuis cette époque ont permis à Qt de s'approcher des demandes actuelles, avec la sortie de Qt Quick par exemple. La base d'utilisateurs de Qt ne cessant de croître, il faut toujours rester proche des besoins nouveaux des développeurs.

Ainsi, pour rester à la pointe de la technologie et un framework leader dans l'industrie en général, Qt doit continuer à se renouveler. Qt 4 a été une évolution, les prochaines versions de Qt devront être dans sa perspective technique. Ces dernières années, de nouvelles fondations ont été esquissées pour la prochaine génération : Qt Quick, QML Scenegraph, Lighthouse, Qt WebKit.

En plus de son orientation vers l'open gouvernance, voici quelques pistes architecturales pour Qt 5.

Objectifs pour Qt 5

Mieux utiliser le GPU, pour obtenir des applications graphiques fluides et performantes, même avec peu de ressources.

Rendre la création d'applications avancées plus facile et plus rapide, avec QML et JavaScript.

Faire en sorte que les applications connectées au Web soient aussi puissantes que possible, notamment au niveau de l'insertion de contenu Web dans des applications Qt.

Réduire la quantité de code requise et sa complexité pour implémenter et maintenir un port.

Qt 4.7 contient encore du vieux code qui empêche certaines évolutions et Qt d'être aussi bon que possible pour la création d'applications next-gen. La plupart du code convient encore, alors que certaines parties bloquent le chemin de Qt 4.x.

Faciliter la migration entre Qt 4 et Qt 5

Avec Qt 5, il est prévu d'effectuer des changements bien précis dans les API pour abandonner certaines limitations du passé. La difficulté du portage de Qt 3 vers Qt 4 ne sera pas réitérée : la compatibilité des sources sera conservée pour la majorité des cas, alors que, à d'autres endroits, il sera nécessaire de la briser.

Il est actuellement envisagé de se focaliser sur un petit ensemble de systèmes supportés (Wayland, X11 pour Linux, en plus de Mac OS X et de Windows), le nombre total de plateformes supportées dépendra de l'activité de la communauté, suivant le principe de l'open gouvernance. Les autres systèmes d'exploitation ne seront pas maintenus par Nokia (principalement les UNIX commerciaux). Le but de Qt 5 est de fournir les meilleures fonctionnalités sur

chaque plateforme, ce qui implique que l'on va d'abord fournir des fonctionnalités plus différenciées entre les OS, tout en offrant une réutilisation du code aussi efficace que possible pour la majorité des plateformes.

Développé ouvertement

Le modèle de développement va complètement changer entre Qt 4 et Qt 5, changement qui se profile depuis déjà longtemps : Qt 4 a principalement été développé par Trolltech puis Nokia, le fruit de ce travail a été publié. Qt 5 se veut développé par la communauté, un projet open source dès le tout début. Il n'y aura pas de différences entre un développeur de chez Nokia et un contributeur externe.

Vision

Qt 5 devrait être la fondation d'une nouvelle manière de développer des applications. En offrant toute la puissance de Qt en C++ natif, le focus sera déplacé vers un modèle, où le C++ est principalement utilisé pour implémenter les fonctionnalités derrière Qt Quick. Qt 5 mettra Qt Quick au centre, sans se couper du code déjà existant, il s'agira plus d'une restructuration. Les applications Qt/C++ traditionnelles vont continuer à fonctionner avec Qt 5, même si quelques changements vont être apportés sur les fondements de la manière de développer des applications.

On devrait s'attendre à ce que toutes les IU soient écrites en QML. JavaScript deviendra un citoyen de première classe de l'environnement Qt : beaucoup de logique, voire toute une application sera écrite en JavaScript, non en C++. Le but est que les développeurs commencent d'abord avec QML et JavaScript pour n'implémenter des fonctionnalités en C++ que lorsque cela est nécessaire. Dans ce cas, toute la puissance des API C++ pourra être utilisée pour implémenter des fonctionnalités plus complexes ou dont l'exécution doit être très rapide.

Le modèle basé sur QWidget sera conservé, les API liées aussi, pour la compatibilité, mais le long terme devrait voir QML comme le futur des interfaces utilisateur pour desktop. La solution retenue sera probablement des composants basés sur QML qui s'intègrent dans le style natif des plateformes desktop.

Les quatre grands changements architecturaux

Les premiers changements sont déjà en cours depuis un certain temps, le travail débute pour le reste. La plupart de ces changements pourraient être effectués avant août.

Changer l'architecture de la pile graphique

Qt Quick et QML Scenegraph seront au centre de la nouvelle architecture. QPainter sera toujours disponible et est très utile pour bien des choses, mais il ne sera plus utilisé pour l'interface principale. Qt aura besoin d'OpenGL (ES) 2.0 pour fonctionner. Les QWidget seront rendus par-dessus le graphe de scène, pas l'inverse comme actuellement.

Qt Components et Qt Mobility feront donc partie intégrante de Qt, plus des modules avec un statut spécial.

Baser tous les ports de Qt sur Lighthouse

Le projet Lighthouse a été lancé pour fournir une meilleure manière d'abstraire l'intégration au système de fenêtrage que ce qui est actuellement disponible. La maturité est proche avec Qt 4.8, son utilisation sera probablement requise pour Qt 5.0.

Utiliser une structure de répertoires modulaire

Beaucoup a déjà été fait ces dernières semaines, le travail est d'ores et déjà visible dans les nouveaux dépôts : [Lien 119](#). Cette modularisation va permettre de faciliter et d'accélérer l'intégration de contributions à Qt.

Séparer les fonctionnalités liées à QWidget dans une bibliothèque distincte

Les classes basées sur QWidget sont actuellement très importantes ; le modèle va cependant changer et les IU seront principalement faites avec QML. Séparer les fonctionnalités basées sur QWidget est donc une manière d'obtenir une architecture propre dans Qt 5.

Beta fin 2011, version finale en 2012

On ne doit pas trop changer dans les fondations de Qt, on veut aussi rendre facile le portage d'applications vers Qt 5 : on ne peut donc pas trop modifier la base de code déjà en place. Beaucoup des changements proposés se basent sur la modularisation de Qt, avec chaque bibliothèque partagée dans son propre répertoire. On devra supprimer quelques API rarement utilisées si elles se révèlent contraignantes à maintenir sans empêcher d'avancer. Du code en version beta devrait être disponible fin 2011, avec une version finale pour 2012.

Retrouvez ce billet sur le blog de Thibaut Cuvelier en ligne : [Lien 120](#)

Les derniers tutoriels et articles

Manipulation d'images avec PyQt

Ce tutoriel est destiné à ceux qui découvrent la bibliothèque Qt et ses nombreuses possibilités dans le domaine de l'imagerie. Nous y aborderons les principes de base en personnalisant une interface et en réalisant une visionneuse d'image.

1. Introduction

La manipulation d'images revêt plusieurs aspects. Tout d'abord l'utilisation d'images sur des constituants de l'interface, appelés widgets, à des fins de personnalisation de cette interface, ensuite les traitements simples des images, classement, visionnage, gestion des métadonnées et quelques manipulations de base telles que pivotement, redimensionnement, etc. Pour terminer, le traitement d'image impliquant une modification de ses propriétés, accès aux pixels, colorimétrie, etc.

Nous verrons dans ce tutoriel les deux premiers aspects de la manipulation d'images.

2. Les types d'images Qt

QPixmap

Une pixmap est optimisée pour l'affichage à l'écran, celle-ci est chargée en mémoire du côté serveur X ou dans la mémoire de la carte graphique. Les données stockées dans l'instance d'un **QPixmap** ne sont qu'une référence à l'image chargée sur le serveur X. Son avantage est de profiter des ressources matérielles (accélération graphique) mais, par contre, cela implique que l'image soit traitée dans la boucle principale du programme et non pas dans un thread séparé.

Une visionneuse utilisera de préférence des **QPixmap**.

QImage

QImage permet un accès direct aux pixels de l'image, est indépendant du matériel, peut être utilisé dans un thread séparé, mais ne profite pas de l'accélération matérielle. Les **QImage** seront préférés dans une application de traitement d'images.

QPicture

Les **QPicture** sont des supports de dessin permettant d'enregistrer une suite de commandes de **QPainter** et de les reproduire sur diverses images.

Utilisé pour la sérialisation de dessins, estampillages, etc.

QBitmap

QBitmap désigne une image monochrome, avec un bit à 0 pour l'arrière-plan (ou pixel transparent) et un bit à 1 pour l'avant-plan (ou pixel opaque).

L'exemple le plus courant étant la création de curseur personnalisé.

QIcon

Les **QIcon** sont des objets dynamiques en ce sens qu'ils sont redimensionnables selon les besoins de l'interface et peuvent revêtir divers états : désactivé, actif, survolé, cliqué, etc. Les **QIcon** sont généralement construits à partir de **QPixmap**.

Nous en verrons des exemples d'utilisation sur des widgets.

3. Les formats d'image

Par défaut, **Qt** peut lire les formats BMP, GIF, ICO, JPEG, JPG, MNG, PBM, PGM, PNG, PPM, SVG, TIF, TIFF, XBM, XPM et peut enregistrer sous les formats BMP, ICO, JPEG, JPG, PNG, PPM, TIF, TIFF, XBM, XPM.

L'ajout d'autres formats doit se faire par plug-in.

Les deux commandes suivantes permettent de savoir quels sont les formats d'image supportés par votre version de **Qt** :

Formats supportés en lecture:

```
for format in
QtGui.QImageReader.supportedImageFormats():
    print format
```

Formats supportés en écriture :

```
for format in
QtGui.QImageWriter.supportedImageFormats():
    print format
```

4. Les modes d'ouverture

Diverses manières permettent d'instancier un objet image, entrons dans le code.

```
# Avec le nom de fichier
image = QtGui.QImage("fichierImage.jpg")
pixmap = QtGui.QPixmap("fichierImage.jpg")

# L'extension peut être séparée
image = QtGui.QImage("fichierImage", "jpg")
pixmap = QtGui.QPixmap("fichierImage", "jpg")

# ou ignorée
image = QtGui.QImage("fichierImage")
pixmap = QtGui.QPixmap("fichierImage")
# dans ce cas Qt déterminera le format par l'en-
tête du fichier.
# ! Lorsque l'extension est donnée, celle-ci doit
être exacte.

# Avec une autre instance d'image
image = QtGui.QImage("fichierImage.jpg")
pixmap = QtGui.QPixmap(image)

# QPixmap possède une méthode .toImage()
image = pixmap.toImage() == image =
QtGui.QImage(pixmap)

# et une méthode .fromImage(image, flag)
pixmap = QtGui.QPixmap.fromImage(image, 0)
```

les différentes valeurs de 'flag' (0 par défaut) peuvent être trouvées ici : [Lien 121](#)

À ce stade, il devient utile de savoir si l'image a été correctement chargée.

En effet, en cas de non-ouverture du fichier, Qt ne retourne pas d'erreur mais crée une "null pixmap", il est donc souvent indispensable de vérifier le bon chargement de l'image au moyen des méthodes **QPixmap.isNull()** et **QImage.isNull()**.

```
if image.isNull():
    print "Image non valide !"
```

5. Les supports

Les images peuvent être appliquées à différents widgets selon qu'elles sont destinées à la décoration de l'interface ou qu'elles sont l'objet même de l'application.

Nous allons voir l'utilisation de ces différents supports par la pratique, pour cela nous allons créer une interface qui nous servira tout au long de ce tutoriel.

imageViewer.py

```
# -*- coding: utf-8 -*-
import sys
import os

from PyQt4 import QtCore, QtGui

class ImageViewer(object):
    def setupUi(self, Viewer):
        Viewer.resize(640, 480)
        Viewer.setWindowTitle(u"Exemples d'usage
d'images")
        self.image_1 = "image1.jpg"
        self.image_2 = "image2.png"
        self.centralwidget =
QtGui.QWidget(Viewer)
        self.gridLayout =
QtGui.QGridLayout(self.centralwidget)
        self.verticalLayout_2 =
QtGui.QVBoxLayout()
        self.horizontalLayout =
QtGui.QHBoxLayout()

        # QLabel
        self.label =
QtGui.QLabel(self.centralwidget)
        self.sizePolicy =
QtGui.QSizePolicy(QtGui.QSizePolicy.Preferred,
QtGui.QSizePolicy.Fixed)
        self.label.setSizePolicy(sizePolicy)
        self.label.setPixmap(QtGui.QPixmap(self.i
mage_1))
        self.label.setScaledContents(True)
        self.horizontalLayout.addWidget(self.labe
l)

        self.verticalLayout = QtGui.QVBoxLayout()
        self.label_cmb =
QtGui.QComboBox(self.centralwidget)
        self.verticalLayout.addWidget(self.label_
cmb)
        spacerItem = QtGui.QSpacerItem(20, 18,
QtGui.QSizePolicy.Minimum,
QtGui.QSizePolicy.Fixed)
        self.verticalLayout.addItem(spacerItem)
        self.horizontalLayout.addLayout(self.vert
icalLayout)
        self.verticalLayout_2.addLayout(self.hori
zontalLayout)
        self.horizontalLayout_2 =
QtGui.QHBoxLayout()

        # QPushButton
        self.pushButton =
QtGui.QPushButton(self.centralwidget)
        self.pushButton.setText("PushButton")
        icon1 = QtGui.QIcon()
        icon1.addPixmap(QtGui.QPixmap(self.image_
2),QtGui.QIcon.Normal,
QtGui.QIc
on.Off)
        self.pushButton.setIcon(icon1)
        self.horizontalLayout_2.addWidget(self.pu
shButton)

        self.push_cmb =
QtGui.QComboBox(self.centralwidget)
        self.horizontalLayout_2.addWidget(self.pu
```



```

sh_cmb)

    # QToolButton
    self.toolButton =
QtGui.QToolButton(self.centralwidget)
    self.toolButton.setText("toolButton")
    self.toolButton.setIcon(icon1)
    self.toolButton.setStyleSheet(QtCore
.Qt.ToolButtonTextBesideIcon)
    self.horizontalLayout_2.addWidget(self.to
olButton)

    self.tool_cmb =
QtGui.QComboBox(self.centralwidget)
    self.horizontalLayout_2.addWidget(self.to
ol_cmb)
    self.verticalLayout_2.addLayout(self.hori
zontalLayout_2)
    self.horizontalLayout_3 =
QtGui.QHBoxLayout()

    # QRadioButton
    self.radioButton =
QtGui.QRadioButton(self.centralwidget)
    self.radioButton.setText("RadioButton")
    self.radioButton.setIcon(icon1)
    self.horizontalLayout_3.addWidget(self.ra
dioButton)

    # QCheckBox
    self.checkBox =
QtGui.QCheckBox(self.centralwidget)
    self.checkBox.setText("CheckBox")
    self.checkBox.setIcon(icon1)
    self.checkBox.setLayoutDirection(QtCore.Q
t.RightToLeft)
    self.horizontalLayout_3.addWidget(self.ch
eckBox)

    spacerItem1 = QtGui.QSpacerItem(40, 20,
QtGui.QSizePolicy.Expanding,
QtGui.QSi
zePolicy.Minimum)
    self.horizontalLayout_3.addItem(spacerIte
m1)

    # Colors comboBox
    self.colors_cmb =
QtGui.QComboBox(self.centralwidget)
    self.horizontalLayout_3.addWidget(self.co
lors_cmb)
    self.verticalLayout_2.addLayout(self.hori
zontalLayout_3)

    # QGraphicsView
    self.vue =
QtGui.QGraphicsView(self.centralwidget)
    self.verticalLayout_2.addWidget(self.vue)

    self.horizontalLayout_4 =
QtGui.QHBoxLayout()
    self.horizontalLayout_4.setObjectName("ho
rizontalLayout_4")
    self.image_btn =
QtGui.QToolButton(self.centralwidget)
    self.image_btn.setText("Image")
    self.image_btn.setObjectName("image_btn")
    self.horizontalLayout_4.addWidget(self.im
age_btn)
    spacerItem2 = QtGui.QSpacerItem(40, 20,
QtGui.QSizePolicy.Expanding,
QtGui.QSi
zePolicy.Minimum)
    self.horizontalLayout_4.addItem(spacerIte

```

```

m2)
    self.verticalLayout_2.addLayout(self.hori
zontalLayout_4)
    self.gridLayout.addLayout(self.verticalLa
yout_2, 0, 0, 1, 1)
    Viewer.setCentralWidget(self.centralwide
t)

    self.populate_combos()

    # Connections
    self.label_cmb.currentIndexChanged.connect
(self.update_label)
    self.push_cmb.currentIndexChanged.connect
(self.update_push_button)
    self.tool_cmb.currentIndexChanged.connect
(self.update_tool_button)

    def populate_combos(self):
        items = ["setScaledContents(True)",
"setScaledContents(False)"]
        self.label_cmb.addItem(items)
        items = ["Modifier...",
"setAutoDefault()", "setDefault()", "setFlat()"]
        self.push_cmb.addItem(items)

        items = ["Modifier...", "setAutoRaise()",
"ToolButtonIconOnly",
"ToolButtonTextOnly",
"ToolButtonTextBesideIcon",
"ToolButtonTextUnderIcon"
, "ToolButtonFollowStyle"]
        self.tool_cmb.addItem(items)

        names = ["Rouge", "Vert", "Bleu"]
        colors = [QtGui.QColor(255, 0, 0, 255),
QtGui.QColor(0, 255, 0, 255),
QtGui.QCo
lor(0, 0, 255, 255)]
        pix = QtGui.QPixmap(QtCore.QSize(30, 10))
        self.colors_cmb.setIconSize(QtCore.QSize
(30, 10))
        for idx, name in enumerate(names):
            pix.fill(colors[idx])
            icon = QtGui.QIcon(pix)
            self.colors_cmb.addItem(icon, name)

    def update_label(self, idx):
        self.label.setScaledContents(not
self.label.hasScaledContents())

    def update_push_button(self, idx):
        if not idx:
            return
        if idx == 1:
            self.pushButton.setAutoDefault(not
self.pushButton.autoDefault())
        elif idx == 2:
            self.pushButton.setDefault(not
self.pushButton.isDefault())
        else:
            self.pushButton.setFlat(not
self.pushButton.isFlat())

    def update_tool_button(self, idx):
        if not idx:
            return
        if idx == 1:
            self.toolButton.setAutoRaise(not
self.toolButton.autoRaise())
        else:
            self.toolButton.setStyleSheet(id

```


x-2)

```
if __name__ == "__main__":
    import sys
    app = QtGui.QApplication(sys.argv)
    Viewer = QtGui.QMainWindow()
    ui = ImageViewer()
    ui.setupUi(Viewer)
    Viewer.show()
    sys.exit(app.exec_())
```

Dans ce code, vous aurez à remplacer dans les lignes 12 et 13 les chemins des images situées sur votre disque. Pour `self.image_2`, choisissez de préférence une image de taille réduite. Une icône fera très bien l'affaire.

Remarque sur les chemins des images :

- soit les images sont dans le même dossier que le script, utilisez "imageX.jpg" ;
- soit elles sont dans un sous-dossier (ex. "medias/") utilisez, "medias/imageX.jpg" ;
- soit elles sont dans le dossier parent (vers le haut) utilisez, "../imageX.jpg" ;
- soit vous ne vous en sortez pas, utilisez le chemin complet ;
- d'autre part, "Couché-de-soleil.jpg" retournera une image nulle, utilisez `u"Couché-de-soleil.jpg"`. (Non, **Qt** n'est pas sensible aux fautes d'orthographe.)

Voyons le code, tout d'abord le **QLabel**, rien de compliqué, un **QPixmap** ou un **QPicture** peuvent lui être appliqué directement. Un argument optionnel permet d'ajuster la taille de l'image à l'espace du **QLabel**, mais cet agrandissement n'est pas proportionnel, l'image peut donc apparaître trop étirée dans un des deux axes.

On ne peut joindre texte et image dans un **QLabel**.

Les **QPushButton** et **QToolButton** demandent la création d'un **QIcon** qui sera appliqué avec la méthode `setIcon(icon)`.

Une option permet de déterminer la taille de l'icône mais sera inopérante si la taille choisie est supérieure à celle du bouton. Il est souvent préférable de laisser **Qt** choisir la taille de l'icône.

Vous remarquerez, dans les combos que ces deux types de boutons n'ont pas les mêmes options d'apparence.

Pour obtenir un résultat comme celui-ci :



on choisira des **QPushButton**.

Les **QRadioButton** et **QCheckBox** demandent aussi un **QIcon** pour l'insertion d'une image. Ici, les options d'apparence se limitent à la direction du widget, c'est-à-dire que la case à cocher peut être positionnée à droite comme dans le cas de la **QCheckBox**.

Pour terminer avec les widgets pouvant être décorés au moyen d'images, nous avons un **QComboBox** où les images sont insérées au moyen de **QIcon**. Ces icônes étant créées à partir de pixmaps dans la dernière partie de la fonction `populate_combos()`.

Toutes les possibilités d'insertion d'image dans tous les widgets possibles ne peuvent être vues ici, mais les méthodes utilisées dans le code devraient permettre de répondre à toutes les situations.

Des personnalisations d'interface plus poussées, couleur de fond, couleur de texte, image de fond, feront appel au `StyleSheet`.

6. Visionnage d'images

Lorsque l'image est l'objet même de l'application, comme dans une visionneuse, divers widgets peuvent servir de support tels que **QFrame**, **QWidget** ou encore **QScrollArea**.

Ces widgets impliquent toutefois que notre code implémente les fonctions de positionnement ou de centrage qui peuvent s'avérer de vrais casse-tête, surtout si l'on désire que la fenêtre soit redimensionnable ou, dans le cas du **QScrollArea** où l'apparition d'une barre de défilement repositionne systématiquement l'image dans le coin supérieur gauche du widget.

Qt nous propose un conteneur beaucoup plus performant pour l'affichage d'image: le **QGraphicsView**. Plus exactement, la paire **QGraphicsView** et **QGraphicsScene**.

Le **QGraphicsView** ou la vue, est l'espace physique dans lequel se positionnera la scène. Ce widget hérite de **QAbstractScrollArea** ce qui nous permettra de profiter de fonctionnalités "ready-to-use" comme nous le verrons avec l'outil panoramique.

Le **QGraphicsScene** ou la scène, est l'espace virtuel dans lequel nous placerons notre image. Cet espace étant virtuel, il ne doit pas obligatoirement être dimensionné, dans ce tuto, nous lui donnerons cependant, les dimensions de l'image à afficher, pour profiter, entre autres, du centrage automatique de l'image dans la scène.

La vue sera donc une fenêtre sur un espace illimité par défaut.

Dans le code, nous avons utilisé la méthode la plus simple pour instancier la vue :

```
self.vue =
QtGui.QGraphicsView(self.centralwidget)
self.verticalLayout_2.addWidget(self.vue)
```

Nous créerons la scène lorsque nous importerons l'image puisque nous avons choisi de lui donner les dimensions de l'image.

La scène aurait pu être créée dès le départ en utilisant la méthode suivante :

```
self.scene = QtGui.QGraphicsScene()
```

```
self.vue = QtGui.QGraphicsView(self.scene)
self.verticalLayout_2.addWidget(self.vue)
```

Remarquez : pas de parent pour la scène, la vue a la scène pour parent et c'est toujours la vue que nous plaçons dans le layout.

Complétons notre code, dans le groupe des connexions (lignes 104-106), ajoutons une ligne :

```
self.image_btn.clicked.connect(self.get_image)
```

ensuite ajoutons la fonction `get_image()` à la fin de notre classe :

```
def get_image(self):
    img =
    unicode(QtGui.QFileDialog.getOpenFileName(Viewer,
        u"Ouverture de fichiers",
        "", "Image Files (*.png *.jpg *.bmp)"))
    if not img:
        return
    self.open_image(img)
```

Cette fonction permettra de choisir une image en cliquant sur le bouton "Image". Remarquez l'utilisation d'Unicode pour éviter le problème d'ouverture cité plus avant.

Terminons en ajoutant les fonctions `open_image()` et `view_current()` :

```
def open_image(self, path):
    w_vue, h_vue = self.vue.width(),
    self.vue.height()
    self.current_image = QtGui.QImage(path)
    self.pixmap = QtGui.QPixmap.fromImage(
        self.current_image.scaled(w_vue, h_vue,
            QtCore.Qt.KeepAspectRatio,
            QtCore.Qt.SmoothTransformation))
    self.view_current()

def view_current(self):
    w_pix, h_pix = self.pixmap.width(),
    self.pixmap.height()
    self.scene = QtGui.QGraphicsScene()
    self.scene.setSceneRect(0, 0, w_pix, h_pix)
    self.scene.addPixmap(self.pixmap)
    self.vue.setScene(self.scene)
```

C'est ici que les choses deviennent intéressantes, voyons ces fonctions en détail.

La fonction `open_image()` :

Nous partons du principe que nous afficherons notre image à la taille de la vue, nous implémenterons un zoom ensuite, donc commençons par extraire la taille de la vue `w_vue` et `h_vue`, respectivement largeur et hauteur.

Créons notre image en tant que `QImage`, celle-ci ne sera jamais affichée mais nous en aurons besoin plus tard.

Nous créons ensuite notre `pixmap` à la dimension de la vue.

Les arguments de la méthode `.scaled()` :

- Les modes de redimensionnement :
 - Qt.IgnoreAspectRatio** le rapport largeur/hauteur de l'image originale ne sera pas respecté, dans la majorité des cas cela entraînera une déformation disgracieuse ;
 - Qt.KeepAspectRatio** le rapport de taille sera respecté et le plus petit agrandissement possible sera utilisé. C'est le mode que nous choisissons ;
 - Qt.KeepAspectRatioByExpanding** le rapport de taille sera respecté et le plus grand agrandissement possible sera utilisé.
- Note : il n'est pas impossible qu'une barre de défilement apparaisse, les dimensions de la vue retournées par Qt peuvent être dépendantes du système.
- Les méthodes de redimensionnement :
 - Qt.FastTransformation** redimensionnement rapide sans antialias ;
 - Qt.SmoothTransformation** redimensionnement avec antialias, le filtre utilisé est de type bilinéaire.

La fonction `view_current()` :

Nous relevons les dimensions de notre `pixmap`, `w_pix` et `h_pix`.

Nousinstancions une scène et nous lui attribuons les dimensions de notre `pixmap`.

Les deux dernières lignes placent notre image dans la scène et ensuite celle-ci dans la vue.

Variante : Si la scène a été créée directement avec la vue, comme indiqué plus haut, la fonction `view_current()` se présentera comme ceci :

```
def view_current(self):
    w_pix, h_pix = self.pixmap.width(),
    self.pixmap.height()
    self.scene.clear()
    self.scene.setSceneRect(0, 0, w_pix, h_pix)
    self.scene.addPixmap(self.pixmap)
```

la ligne `'self.vue.setScene(self.scene)'` ne se justifiant plus ici.

Afin de comprendre pourquoi nous imposons à la scène les dimensions de l'image, je vous propose de tester le code avec des dimensions très différentes de l'image.

ex. si l'image mesure 2500x1800 px, essayez ceci :

```
self.scene.setSceneRect(0, 0, 6000, 4000)
# et ceci :
self.scene.setSceneRect(0, 0, 500, 300)
```

Testez le code et vous constaterez que l'image n'est plus centrée.

Remettez le code dans l'état initial.

```
self.scene.setSceneRect(0, 0, w_pix, h_pix)
```

Comme nous ne sommes jamais satisfaits, implémentons

un zoom. Tout d'abord, il faut mettre la vue à l'écoute de la roulette de la souris. Dans la définition de la vue (ligne 84), rajoutons l'évènement wheelEvent.

Nous devons avoir ceci :

```
# QGraphicsView
self.vue =
QtGui.QGraphicsView(self.centralwidget)
self.vue.wheelEvent = self.wheel_event
self.verticalLayout_2.addWidget(self.vue)
```

et, à la fin de notre code, ajoutons ces fonctions :

```
def wheel_event (self, event):
    steps = event.delta() / 120.0
    self.zoom(steps)
    event.accept()

def zoom(self, step):
    w_pix, h_pix = self.pixmap.width(),
self.pixmap.height()
    w, h = w_pix * (1 + 0.1*step), h_pix * (1 +
0.1*step)
    self.pixmap = QtGui.QPixmap.fromImage(
        self.current_image.scaled(w, h,
            QtCore.Qt.KeepAspectRatio,
            QtCore.Qt.FastTransformation))
    self.view_current()
```

Voyons cela en détail.

La fonction wheel_event() :

event.delta() nous retourne la rotation de la roulette en 1/8 de degré, la plupart des souris étant crantées tous les 15° nous divisons par 120 pour obtenir le nombre de crans, plus évident pour l'utilisateur. Autrement dit, "chaque cran = un niveau de zoom" ; event.delta() sera positif en cas de rotation vers l'avant de la souris et négatif pour une rotation vers l'utilisateur.

La fonction zoom() :

Récupérons tout d'abord les dimensions de notre pixmap w_pix et h_pix.

Appliquons à ces dimensions notre facteur d'agrandissement comme ceci :

supposons une largeur de pixmap de 600 pxl, un pas de zoom de 0.1 (valeur que vous décidez vous-même) et deux steps (crans de souris), nouvelle largeur = 600 * (1 + 0.1 * 2)

Maintenant, c'est ici que nous découvrons l'intérêt d'avoir conservé une instance de notre image originale, en effet, nous ne pouvons nous permettre de recharger l'image depuis le disque pour chaque saut de zoom et d'autre part, si nous redimensionnons directement notre pixmap nous allons voir celle-ci se dégrader de façon exponentielle, chaque redimensionnement amplifiant les erreurs du précédent.

Pour vous en convaincre, modifiez le code comme ceci : dans la fonction zoom() remplacez

```
self.pixmap = QtGui.QPixmap.fromImage(
    self.current_image.scaled(w, h,
        QtCore.Qt.KeepAspectRatio,
        QtCore.Qt.FastTransformation))
```

par ceci :

```
self.pixmap = self.pixmap.scaled(w, h,
    QtCore.Qt.KeepAspectRatio,
    QtCore.Qt.FastTransformation)
```

Testez le code, la dégradation de l'image apparaît rapidement.

Il nous manque encore une chose, c'est de pouvoir déplacer l'image "à la main" lorsque celle-ci est plus grande que la vue.

Retournons dans notre code et ajoutons une ligne à la définition de notre vue pour obtenir ceci :

```
# QGraphicsView
self.vue = QtGui.QGraphicsView(
    self.centralwidget)
self.vue.setDragMode(
    QtGui.QGraphicsView.ScrollHandDrag)
self.vue.wheelEvent = self.wheel_event
self.verticalLayout_2.addWidget(self.vue)
```

C'est tout ce dont nous avons besoin pour notre outil panoramique. Les options de la méthode setDragMode() sont :

- **QGraphicsView.noDrag** supprimer toute fonctionnalité préalablement définie ;
- **QGraphicsView.ScrollHandDrag** déplacer l'image avec la souris ;
- **QGraphicsView.RubberBandDrag** tirer un rectangle de sélection.

Relancez le script, zoomez et déplacez l'image, nous avons construit une modeste visionneuse.

7. Conclusions

Nous avons vu ici, les outils élémentaires de Qt permettant une manipulation simple d'images avec peu de lignes de code ainsi que les bases de la personnalisation d'interface.

Toutefois, ceci n'est qu'un aperçu des possibilités de Qt, des techniques plus poussées deviennent accessibles par une étude des classes suivantes :

QtGui.QImage ;
QtGui.QPixmap ;
QtGui.QImageReader ;
QtGui.QImageWriter ;
QtGui.QIcon ;
et **QtGui.QMatrix**.

8. Liens

Pour accéder à la documentation de l'ensemble des classes :

PyQt : [Lien 122](#) ;
PySide : [Lien 123](#).

Et bien sûr, n'hésitez pas à utiliser le forum pour commenter cet article ou poser vos questions : Forum PyQt PySide ([Lien 124](#)).

Retrouvez l'article de Vincent Vande Vyvre en ligne : [Lien 125](#)

Visual Basic

Les derniers tutoriels et articles

xGUICOM : composant COM (GUI) portable pour langage Active Scripting (VBScript/JScript)

Oubliez les applications HTML (HTA) pour définir les interfaces graphiques de vos scripts VBS/JS. xGUICOM est un composant WSC qui fournit une interface graphique aux langages de script capables d'exploiter les composants COM. Sa syntaxe est très simple et familière aux utilisateurs du VBA.

Le code du composant étant lui-même écrit en VBScript, xGUICOM peut être adapté et complété sans difficulté. Il est en outre totalement autonome puisqu'il ne requiert aucune inscription dans la base de registre. Enfin, un éditeur visuel de ressources comme Resource Hacker ([Lien 126](#)) permet de concevoir rapidement et sans peine les boîtes de dialogue de cette interface.

1. Introduction

Microsoft n'a pas jugé utile de doter ses langages VBScript/Jscript de la faculté de gérer une interface graphique sérieuse, y compris lorsque ceux-ci sont exécutés dans le contexte *Windows Script Host*. À ceux qui souhaitent, malgré tout, pouvoir en bénéficier, l'éditeur propose une solution à partir de pages DHTML interprétées par le moteur de rendu d'Internet Explorer.

Cette solution est clairement un pis-aller qui présente, à mon sens, deux inconvénients majeurs :

- le rendu final est susceptible de varier en fonction des évolutions qui sont ou seront apportées au moteur de rendu d'un produit aussi stratégique qu'IE ;
- l'interprétation effective de la page n'est pas vraiment conforme à la logique de la programmation impérative suivie par le langage lui-même et nécessite une écriture souvent verbeuse.

Pour y remédier, il existe déjà sur le net plusieurs composants COM gratuits qui permettent la création et la gestion d'une interface graphique :

- WshDialog écrit en VB6 qui utilise plusieurs autres composants *ActiveX* ce qui ne simplifie malheureusement pas le déploiement : [Lien 127](#) ;
- Quick prompts, simple, sans dépendances particulières, mais limité par sa licence à un usage non commercial : [Lien 128](#) ;
- WindowSystemObject (WSO) le composant de loin le plus puissant et complet (presque trop) offrant notamment la possibilité d'insérer les composants *ActiveX* visuels de son choix : [Lien 129](#). La documentation est rigoureuse et complète mais uniquement en anglais (ou russe).

Pourquoi un "yet another" alors qu'il existe une telle offre ? Tout simplement parce que ces solutions présentent toutes les mêmes faiblesses :

- obligation préalable d'inscrire le composant dans la base de registre, sauf à rédiger un *manifest*, mais la complexité de l'interface ne facilite pas son écriture ; (pour plus de détails, voir cet article : [Lien 130](#))
- la disposition des contrôles dans la boîte de dialogue se fait à l'aveuglette ce qui rend très pénible la création des masques même si le composant *QuickPrompts* propose un "automatic form layout" utile pour les dialogues simples.

xGUICOM évite ces deux écueils puisqu'il n'a pas besoin d'être obligatoirement enregistré s'il est référencé par un *moniker*. Par ailleurs, si les méthodes traditionnelles de création et d'ajout de contrôles sont présentes dans le composant, celui-ci a la capacité d'exploiter des ressources binaires créées par des éditeurs externes, ce qui autorise une création confortable et visuelle de la boîte de dialogue.

Bien évidemment, il ne s'agit pas de rivaliser avec des bibliothèques comme **QT** ou **wxWidgets** mais plus modestement de permettre l'écriture rapide d'une interface efficace pour des scripts ponctuels. Cette version 1.0 supporte néanmoins une quinzaine de contrôles de base dont vous trouverez la description détaillée ci-dessous.

2. Présentation de xGUICOM

xGUICOM se présente sous la forme d'un *Windows Script Component* (WSC), c'est-à-dire un composant COM écrit dans un langage de script interprété, en l'occurrence le VBScript. C'est un simple fichier texte dénommé **xGuiCom.wsc**. Cela permet à xGUICOM de proposer plusieurs niveaux d'utilisation :

- le niveau (très) basique qui consiste à traiter le dialogue créé par xGUICOM comme une *MsgBox* géante dont il reproduit le comportement sans se soucier de gérer les événements de celui-ci ;
- le niveau intermédiaire qui s'attache à gérer les événements déclenchés par les actions de l'utilisateur et de récupérer les valeurs des contrôles avant la fermeture du dialogue ;
- le niveau avancé qui ne concerne *a priori* qu'une petite minorité d'utilisateurs et qui consiste à modifier directement le contenu du composant, soit pour le simplifier, soit au contraire pour y ajouter des fonctionnalités comme de nouveaux événements. Les sections 5 et 8 s'adressent tout particulièrement à ces utilisateurs avancés ;

xGUICOM crée ses boîtes de dialogues en faisant appel à l'API Win32 de Windows. La gestion des DLL est nativement impossible en VBScript et il est donc nécessaire de recourir à un composant tiers pour y parvenir (*DynamicWrapperX*). Afin d'assurer son déploiement sans inscription dans la base de registre, le code du composant reprend la quasi-totalité du code décrit dans un précédent article : [Lien 131](#). Seule la fonction de décodage *Base64* a

été légèrement modifiée pour pouvoir renvoyer une variable chaîne et pas seulement un fichier.

La section 3 ([Lien 132](#)) contient la documentation de référence pour la mise en œuvre pratique du composant. La section 4 ([Lien 133](#)) contient un bref tutoriel sur la manière d'utiliser un éditeur de ressources pour créer rapidement des boîtes de dialogue. Enfin, les sections 5 ([Lien 134](#)) et 8 ([Lien 135](#)) ne présentent d'intérêt que pour les développeurs qui voudront adapter le composant à leur convenance.

Pour des raisons d'encombrement, le code des fichiers binaires n'a pas été reproduit dans le script ci-dessous mais il est présent dans l'exemplaire téléchargeable dont le lien se trouve à la section 6 : [Lien 136](#).

3. Référence

3.1. Objet

La conception d'**xGUICOM** s'écarte volontairement du paradigme POO pour n'en conserver que le strict minimum. Ainsi, le modèle objet du composant est réduit à un objet racine *Dialog* doté de neuf méthodes et de sept événements.

3.1.1. Dialog

L'instanciation de l'objet se fait au moyen du *moniker* de composant, c'est-à-dire un identifiant chaîne unique composé de "script:" suivi du nom complet du fichier qui peut être une URL et terminé éventuellement par "#nomducomposant" si le fichier .wsc contient plusieurs composants distincts. La connexion sortante de l'objet n'est pas supportée dans cette configuration. Elle doit donc être réalisée séparément au moyen de la fonction *ConnectObject* afin de pouvoir gérer les événements dans le script client.

L'instanciation correcte de l'objet est ainsi assurée en **VBScript** de la façon suivante (*oDlg* est le nom de l'instance de l'objet *Dialog* et "ev_" le nom du préfixe) ou son équivalent dans un autre langage :

```
Set oDlg = GetObject("script:" &
Left(WScript.ScriptFullName, Len(WScript.ScriptFull
Name) - Len(WScript.ScriptName)) &
"xGuiCom.wsc")
WScript.ConnectObject oDlg, "ev_"
```

3.2. Méthodes

Les paramètres obligatoires sont mentionnés en gras. Les autres paramètres ne peuvent être omis et doivent être au moins représentés soit par une chaîne nulle, soit par la valeur zéro.

3.2.1. CreateForm(**sCaption**, **lLeft**, **lTop**, **lWidth**, **lHeight**)

Crée une boîte de dialogue vierge sans l'afficher.

sCaption : chaîne - titre de la boîte.

lLeft, **lTop** : numérique - coordonnées relatives de l'angle haut gauche du dialogue (twips).

lWidth, **lHeight** : numérique - largeur et hauteur du dialogue (twips).

Valeur retournée : numérique - ID du dialogue créé (incrément automatique à partir de zéro). En cas d'erreur, renvoie -1.

Note :

La boîte créée devient le dialogue courant. Toute méthode *AddControl* appelée ultérieurement s'appliquera automatiquement à ce dialogue.

3.2.2. AddControl(**iID**, **sClass**, **lLeft**, **lTop**, **lWidth**, **lHeight**, **sData**, **iStyle**)

Ajoute un contrôle à la boîte de dialogue courante.

iID : numérique - identifiant unique entre 1..65535.

sClass : chaîne - nom de classe du contrôle (casse indifférente).

Noms supportés : *commandbutton* ;

optionbutton ;

checkbox ;

frame ;

edit ;

edpswd ;

memo ;

label ;

image ;

listbox ;

scrollbar ;

combobox ;

hotkey ;

ipcontrol ;

progressbar ;

filedlgbox.

lLeft, **lTop** : numérique - coordonnées relatives de l'angle haut gauche du contrôle (twips).

lWidth, **lHeight** : numérique - largeur et hauteur du contrôle (twips).

sData : chaîne - données initiales du contrôle.

iStyle : numérique - style propre du contrôle.

Valeur retournée : booléenne - *True* en cas de succès, *False* si l'ajout a échoué.

Note :

Selon les contrôles concernés, la syntaxe de *sData* sera différente :

commandbutton, *optionbutton*, *checkbox*, *frame*, *label* : légende du contrôle, chaîne libre.

edit, *edpswd*, *memo* : contenu initial du contrôle, chaîne libre.

listbox, *combobox* : liste des items du contrôle, chaque item est séparé par le caractère |. (ex : "poire|pomme|cerise")

hotkey : valeur initiale du contrôle, syntaxe identique à celle de la fonction **VbScript SendKeys**. (ex : "{^%F2}" équivaut à Ctrl + Alt + F2)

ipcontrol : valeur initiale du contrôle, chaîne conforme au format IPv4. (ex : "120.52.255.255")

filedlgbox : paramètres de la boîte de dialogue fichier séparés par le caractère | selon le format "caption|type|iodata|filename|initialdir|titre".

caption : chaîne, libellé du bouton d'ouverture du dialogue fichier.

type : 0 = *OpenFileDialogBox*, 1 = *SaveFileDialogBox*.

iodata : nom de variable ou ID du contrôle contenant les données en entrée (*open*) ou en sortie (*save*).

filename : chaîne, nom de variable ou ID du contrôle contenant le nom par défaut du fichier.

initialdir : chaîne, nom de variable ou ID du contrôle contenant le répertoire par défaut.

titre : chaîne, nom de variable ou ID du contrôle contenant le titre du dialogue.

image, scrollbar, progressbar : non exploité, peut être une chaîne nulle.

Par défaut, il est assigné un style standard pour chaque contrôle et *iStyle* est alors égal à zéro. Le contrôle *listbox* est une liste classée et doté d'un ascenseur vertical, le contrôle *combobox* est du type *dropdown* avec ascenseur vertical. Il est possible d'assigner une combinaison différente de styles au contrôle en donnant au paramètre *iStyle* une valeur supérieure à zéro.

Néanmoins, *iStyle* peut avoir une signification particulière pour certains contrôles. (voir infra)

3.2.3. LoadLayoutFromRes(sData)

Crée une boîte de dialogue avec ses contrôles à partir de ressources binaires issues d'un éditeur externe.

sData : chaîne - nom de constante ou nom de fichier contenant les ressources. S'il s'agit d'une constante du script, les ressources sont codées en *Base64*.

Valeur retournée : numérique - ID du dialogue créé (incrément automatique à partir de zéro). En cas d'erreur, renvoie un nombre négatif :

-1 : objets *ADO.Stream* et *FileSystem* absents si *sData* est un nom de fichier existant ;

-2 : fichier et constante inexistant ;

-3 : format fichier binaire incorrect ;

-4 : erreur initialisation.

Note :

La boîte créée devient le dialogue courant. Toute méthode *AddControl* appelée ultérieurement s'appliquera automatiquement à ce dialogue.

3.2.4. Show(iIDD, bOnTaskBar)

Affiche la boîte de dialogue portant l'ID correspondant.

iIDD : numérique - identifiant de la boîte de dialogue préalablement créée par la méthode *CreateForm* ou *LoadLayoutFromRes*.

bOnTaskBar : booléen - flag d'affichage du script dans la barre des tâches, *True* = affiché *False* = non affiché, par défaut *False*.

Valeur retournée : numérique

-1 : erreur ;

0 : fermeture bouton système ;

1 : fermeture bouton par défaut ;

2 : fermeture touche *Esc* ou bouton "*Annuler*";

3..7 : ID du bouton contrôle actionné.

Note :

Les valeurs renvoyées reproduisent le comportement de la boîte de dialogue native de *VbScript* appelée par la fonction *MsgBox*. Cela implique qu'il existe des valeurs réservées pour le choix des ID (voir infra).

3.2.5. GetValueFromID(iID, hWndDlg)

Lit la valeur du contrôle *iID* contenu dans le dialogue d'handle *hWndDlg*.

iID : numérique - identifiant du contrôle ;

hWndDlg : numérique - handle de la boîte de dialogue ;

Valeur retournée : dépend de la nature du contrôle (voir infra pour chaque contrôle). Si erreur renvoie *VbNull*.

Note : cette méthode permet notamment lors de l'évènement *Close* de récupérer les valeurs de certains contrôles du dialogue. S'applique aux contrôles *OptionButton, CheckBox, Frame, Edit, EdPswd, Memo, ListBox, ComboBox, Image, HotKey* et *IPControl*.

3.2.6. SetValueFromID(iID, hWndDlg, vData)

Assigne la valeur *sData* au contrôle *iID* contenu dans le dialogue d'handle *hWndDlg*.

iID : numérique - identifiant du contrôle ;

hWndDlg : numérique - handle de la boîte de dialogue ;

vData : chaîne/numérique - valeur assignée au contrôle, elle dépend de sa nature (voir infra pour chaque contrôle) ;

Valeur retournée : *True* ou *False* en cas d'erreur.

Note : cette méthode est particulièrement utile lors des évènements *Create, Click* ou *Change* pour mettre à jour certains contrôles du dialogue. S'applique aux contrôles *OptionButton, CheckBox, Frame, Edit, EdPswd, Memo, ListBox, ComboBox, Image, HotKey* et *IPControl*.

3.2.7. AddItem(iID, hWndDlg, sData, iIndex)

Ajoute un item *sData* à un contrôle *ListBox* ou *ComboBox* au rang *iIndex*.

Si *iIndex* = -1, l'item sera placé en fin de liste.

iID : numérique - identifiant du contrôle ;

hWndDlg : numérique - handle de la boîte de dialogue ;

sData : chaîne - contenu de l'item ;

iIndex : rang d'insertion de l'item (base 0) ;

Valeur retournée : rang effectif de l'item inséré ou -1 si erreur.

3.2.8. RemoveItem(iID, hWndDlg, iIndex)

Retire un item *sData* de rang *iIndex* d'un contrôle *ListBox* ou *ComboBox*.

iID : numérique - identifiant du contrôle ;

hWndDlg : numérique - handle de la boîte de dialogue ;

iIndex : rang de l'item à supprimer (base 0) ;

Valeur retournée : nombre d'items après suppression ou -1 si erreur.

3.2.9. BinToB64(sFileName)

Convertit un fichier binaire en une chaîne au format *B64*.

sFileName : nom complet du fichier binaire cible ;

Valeur retournée : chaîne au format *B64* contenant les données binaires.

Note : cette méthode n'a pas de rapport direct avec la gestion de la boîte de dialogue. Sa présence permet de simplifier l'intégration d'un dialogue sous forme binaire au sein d'un script client (voir l'exemple fourni *BuildConstFromBin.vbs*).

3.3. Événements

3.3.1. Launch

Éventuellement déclenché lors du premier appel des méthodes *CreateForm* et *LoadLayoutFromRes*. Il ne sera

pas activé si le composant COM *DynamicWapperX* est déjà enregistré dans la base de registre.

Si ce composant *DynamicWapperX* n'a pas été enregistré, l'évènement doit alors impérativement être géré par le code **VBScript** suivant ("ev_" est un préfixe arbitraire) ou son équivalent dans un autre langage :

```
Sub ev_Launch
    Set oShell = CreateObject("WScript.Shell")
    oShell.Run "WScript.exe " & Chr(34) &
WScript.ScriptFullName & Chr(34)
    WScript.Quit
End Sub
```

3.3.2. Create(iIDD, hWndDlg)

Déclenché lors de la création effective de la boîte de dialogue par Windows. Les contrôles existent et il est donc possible de les initialiser avec des valeurs différentes de celles définies par *AddControl* ou par les données de ressources binaires.

iIDD : numérique - identifiant unique du dialogue de zéro à n.

hWndDlg : numérique - handle de la fenêtre de dialogue.

Attention, seule la méthode *Show* crée une boîte de dialogue. Comme son nom ne l'indique pas, la méthode *CreateForm* construit seulement une structure de données complétée par *AddControl* qui sera ensuite exploitée par *Show*.

3.3.3. Close(iIDD, hWndDlg, iID)

Déclenché à la fermeture de la boîte de dialogue avant sa destruction. Comme la fenêtre existe encore à ce stade, c'est l'endroit idéal pour lire les valeurs utiles du dialogue entrées ou sélectionnées par l'utilisateur.

iIDD : numérique - identifiant unique du dialogue de zéro à n.

hWndDlg : numérique - handle de la fenêtre de dialogue.

iID : numérique - identifiant unique du contrôle qui est à l'origine de la fermeture. Si c'est le bouton de la barre système, iID sera égal à zéro. Cette valeur est également retournée par la méthode *Show*.

3.3.4. Click(iIDD, hWndDlg, iID)

Déclenché par l'action de l'utilisateur sur un contrôle gérant cet évènement. Les contrôles concernés sont *CommandButton*, *OptionButton*, *CheckBox* et *Frame*.

iIDD : numérique - identifiant unique du dialogue de zéro à n.

hWndDlg : numérique - handle de la fenêtre de dialogue.

iID : numérique - identifiant unique du contrôle qui est à l'origine de l'évènement.

3.3.5. Change(iIDD, hWndDlg, iID)

Déclenché par un changement de la valeur d'un contrôle gérant cet évènement. Les contrôles concernés sont *Edit*, *EdPswd*, *Memo* et *ListBox*.

iIDD : numérique - identifiant unique du dialogue de zéro à n.

hWndDlg : numérique - handle de la fenêtre de dialogue.

iID : numérique - identifiant unique du contrôle qui est à l'origine de l'évènement.

3.3.6. Open(iIDD, hWndDlg, iID, sFileName)

Déclenché après la fermeture du dialogue système permettant de sélectionner un fichier à ouvrir.

iIDD : numérique - identifiant unique du dialogue de zéro à n.

hWndDlg : numérique - handle de la fenêtre de dialogue.

iID : numérique - identifiant unique du contrôle qui est à l'origine de l'évènement.

sFileName : chaîne - nom complet du fichier qui a été sélectionné par l'utilisateur.

3.3.7. Save(iIDD, hWndDlg, iID, sFileName)

Déclenché après la fermeture du dialogue système permettant de sélectionner un nom de fichier à sauvegarder.

iIDD : numérique - identifiant unique du dialogue de zéro à n.

hWndDlg : numérique - handle de la fenêtre de dialogue.

iID : numérique - identifiant unique du contrôle qui est à l'origine de l'évènement.

sFileName : chaîne - nom complet du fichier qui a été sélectionné par l'utilisateur.

3.4. Contrôles

Les contrôles décrits dans la présente section sont ceux reconnus par la méthode *AddControl*. Il en existe d'autres qui peuvent être créés par un éditeur de ressources et qui seront reconnus par la méthode *LoadLayoutFromRes*. Toutefois, le code du composant devra être complété pour permettre leur gestion spécifique.

3.4.1. CommandButton

Bouton de commande destiné à être actionné ou cliqué par l'utilisateur.

iID : choix libre entre 1 et 65535.

Les valeurs 1 à 7 permettent de créer des boutons de fermeture et d'émuler ainsi le comportement de la fonction *MsgBox*. Les boutons créés avec des valeurs supérieures à 7 ne déclencheront que l'évènement *Click*.

sData : chaîne - légende du bouton de commande, le raccourci clavier est précédé du caractère &.

iStyle : une valeur égale à zéro assigne un style standard au bouton. Une valeur égale à un définit le bouton par défaut. Pour modifier en profondeur le style du contrôle, se reporter à la documentation MSDN Library.

3.4.2. OptionButton

Bouton radio permettant de définir des options alternatives. À utiliser par groupe de deux minimum.

iID : choix libre entre 8 et 65535.

sData : chaîne - légende du bouton radio, le raccourci clavier est précédé du caractère &.

iStyle : une valeur égale à un initialise le bouton en lui donnant le statut sélectionné. Pour modifier en profondeur le style du contrôle, se reporter à la documentation MSDN Library.

GetValueFromID : numérique - lit le statut du bouton - 0 : non pressé, 1 : pressé.

vData : numérique - fixe le statut du bouton - 0 : non

pressé, 1 : pressé.

Il est nécessaire de regrouper ces contrôles dans un contrôle *Frame*. Il suffit de les ajouter immédiatement après un *Frame*.

3.4.3. CheckBox

Case à cocher permettant de sélectionner des paramètres.

iID : choix libre entre 8 et 65535.

sData : chaîne - légende de la case à cocher, le raccourci clavier est précédé du caractère &.

iStyle : une valeur égale à un initialise la case en lui donnant le statut coché. Pour modifier en profondeur le style du contrôle, se reporter à la documentation MSDN Library.

GetValueFromID : numérique - lit le statut du bouton - 0 : non pressé, 1 : pressé.

vData : numérique - fixe le statut du bouton - 0 : non pressé, 1 : pressé.

3.4.4. Frame

Cadre permettant de regrouper visuellement des contrôles dans le dialogue. Ce contrôle est indispensable pour gérer les groupes de contrôles *OptionButton*.

iID : choix libre entre 8 et 65535.

sData : chaîne - légende du cadre (facultatif), le raccourci clavier est précédé du caractère &.

iStyle : par défaut zéro, pour modifier en profondeur le style du contrôle, se reporter à la documentation MSDN Library.

3.4.5. Edit

Contrôle permettant l'édition de valeurs alphanumériques sur une seule ligne.

iID : choix libre entre 8 et 65535.

sData : chaîne - valeur initiale du contrôle.

iStyle : par défaut zéro, pour modifier en profondeur le style du contrôle, se reporter à la documentation MSDN Library.

GetValueFromID : chaîne - lit le contenu du champ.

vData : chaîne - remplace le contenu du champ par *vData*.

3.4.6. EdPswd

Contrôle identique au précédent à l'exception des caractères masqués, ce qui le destine à la saisie des mots de passe.

iID : choix libre entre 8 et 65535.

sData : chaîne - valeur initiale du contrôle.

iStyle : par défaut zéro, pour modifier en profondeur le style du contrôle, se reporter à la documentation MSDN Library.

GetValueFromID : chaîne - lit le contenu du champ.

vData : chaîne - remplace le contenu du champ par *vData*.

3.4.7. Memo

Contrôle identique au contrôle *Edit* à l'exception de la saisie qui peut s'effectuer sur plusieurs lignes.

iID : choix libre entre 8 et 65535.

sData : chaîne - valeur initiale du contrôle.

iStyle : par défaut zéro, pour modifier en profondeur le style du contrôle, se reporter à la documentation MSDN Library.

GetValueFromID : chaîne - lit la première ligne du champ.

vData : chaîne - remplace la première ligne du champ par *vData*.

3.4.8. Label

Texte statique permettant notamment de placer une légende pour les contrôles qui ne le gèrent pas.

iID : choix libre entre 8 et 65535.

sData : chaîne - contenu du contrôle, le raccourci clavier est précédé du caractère &.

iStyle : par défaut zéro, pour modifier en profondeur le style du contrôle, se reporter à la documentation MSDN Library.

3.4.9. Image

Affiche une image stockée dans un fichier au format bmp.

iID : choix libre entre 8 et 65535.

sData : chaîne - nom complet du fichier contenant l'image.

iStyle : par défaut zéro, pour modifier en profondeur le style du contrôle, se reporter à la documentation MSDN Library.

GetValueFromID : chaîne - renvoie le nom complet du fichier bmp affiché.

vData : chaîne - nom complet du nouveau fichier bmp à afficher.

3.4.10. ListBox

Contrôle affichant une liste d'items et permettant leur sélection par l'utilisateur. Suivant les dimensions de ce contrôle, une partie seulement de la liste sera affichée à la fois.

iID : choix libre entre 8 et 65535.

sData : chaîne - liste des items séparés par le caractère |.

iStyle : par défaut zéro, la liste est ordonnée et n'autorise qu'une seule sélection, pour modifier en profondeur son style, se reporter à la documentation MSDN Library.

GetValueFromID : chaîne - lit l'item sélectionné.

vData : numérique - place le curseur sur l'item d'indice *vData* (base zéro).

3.4.11. ScrollBar

Contrôle affichant un curseur coulissant dans une barre afin de permettre d'afficher l'état d'une action ou de laisser l'utilisateur agir par le biais d'une souris.

iID : choix libre entre 8 et 65535.

sData : numérique - .

iStyle : une valeur égale à zéro définit une barre horizontale, une valeur égale à un, une barre verticale. Pour modifier en profondeur son style, se reporter à la documentation MSDN Library.

Sa mise en œuvre suppose une gestion des notifications envoyées par le contrôle. Cette gestion dépend étroitement des objectifs recherchés. Aussi ce composant ne propose pas de code générique et il appartient à l'utilisateur de l'écrire.

3.4.12. ComboBox

Contrôle affichant une liste déroulante d'items et l'affichage particulier de l'item sélectionné.

iID : choix libre entre 8 et 65535.

sData : chaîne - liste des items séparés par le caractère |.

iStyle : par défaut zéro, la liste n'est pas modifiable et déroulante, pour modifier en profondeur son style, se reporter à la documentation MSDN Library.

GetValueFromID : chaîne - lit l'item sélectionné.

vData : numérique - place le curseur sur l'item d'indice *vData* (base zéro).

3.4.13. HotKey

Contrôle affichant un raccourci clavier et facilitant son édition directement par l'utilisateur.

iID : choix libre entre 8 et 65535.

sData : chaîne - raccourci clavier selon le format reconnu par la fonction *SendKeys*.

iStyle : par défaut zéro, pour modifier en profondeur son style, se reporter à la documentation MSDN Library.

GetValueFromID : chaîne - lit le raccourci du champ.

vData : chaîne - remplace le raccourci existant par celui défini dans *vData* (format syntaxe *SendKeys()*).

3.4.14. IPControl

Contrôle affichant une adresse selon le format IPv4 et facilitant son édition par l'utilisateur.

iID : choix libre entre 8 et 65535.

sData : chaîne - adresse IPv4 conforme soit quatre groupes de trois chiffres séparés par un point.

iStyle : par défaut zéro, pour modifier en profondeur son style, se reporter à la documentation MSDN Library.

GetValueFromID : chaîne - lit le contenu du champ.

vData : chaîne - remplace le contenu du champ par *vData*.

3.4.15. ProgressBar

Contrôle permettant d'afficher visuellement la progression d'une opération normalement invisible.

iID : choix libre entre 8 et 65535.

sData : numérique - (non géré).

iStyle : par défaut zéro, pour modifier en profondeur son style, se reporter à la documentation MSDN Library.

Sa mise en œuvre requiert la création d'un *timer* gérant l'état d'avancement de l'opération décrite par le contrôle. La nature de ce *timer* étant étroitement liée à ladite opération, ce composant ne propose pas de code générique et il appartient à l'utilisateur de l'écrire.

3.4.16. FileDlgBox

Contrôle affichant un bouton de commande destiné à être actionné ou cliqué par l'utilisateur. Lorsqu'il est sollicité, le contrôle affiche un dialogue système, soit pour ouvrir un fichier, soit pour le sauvegarder.

iID : choix libre entre 8 et 65535.

sData : chaîne - paramètres de la boîte de dialogue fichier séparés par le caractère | selon le format "*caption|type|iodata|filename|initialdir|titre*".

caption : chaîne, libellé du bouton d'ouverture du

dialogue fichier.

type : 0 = *OpenFileDialogBox*, 1 = *SaveFileDialogBox*.

iodata : nom de variable ou ID du contrôle contenant les données en entrée (*open*) ou en sortie (*save*).

filename : chaîne, nom de variable ou ID du contrôle contenant le nom par défaut du fichier.

initialdir : chaîne, nom de variable ou ID du contrôle contenant le répertoire par défaut.

titre : chaîne, nom de variable ou ID du contrôle contenant le titre du dialogue.

iStyle : une valeur égale à zéro assigne un style standard au bouton. Une valeur égale à un définit le bouton par défaut. Pour modifier en profondeur le style du contrôle, se reporter à la documentation MSDN Library.

4. Création de boîtes de dialogue avec l'éditeur de ressources

La possibilité de définir visuellement les boîtes de dialogue est un avantage notable de **xGUICOM**. Pour illustrer cette fonctionnalité, j'ai retenu l'éditeur **Resource Hacker** bien connu de tous les bidouilleurs. Comme il ne permet pas de créer un fichier ressource *.res* mais seulement de modifier un fichier existant, le pack à télécharger contient un fichier *Dialog.res* comportant une *dialogbox* comme seule ressource. L'ajout et le positionnement des contrôles se font à la souris de façon intuitive. Cet article n'est pas une initiation à ce logiciel et il convient de se reporter à l'aide pour plus de détails. L'utilisateur constatera que **Resource Hacker** propose sur sa palette plusieurs contrôles qui ne figurent pas dans la liste de **xGUICOM** comme *SysTreeView32*, *SysMonthCal32* ou encore *SysTabControl32*. Ces contrôles seront acceptés et affichés par **xGUICOM** mais il appartient à l'utilisateur de compléter le code de la *DialogProc* afin de gérer les notifications et messages qui leur sont spécifiques.

Après avoir défini les contrôles **et leurs ID**, la *dialogbox* peut alors être compilée puis enregistrée dans un fichier sous forme binaire **et non comme une ressource** (cf. menu Action). Le nom du fichier est ensuite fourni comme paramètre de la méthode *LoadLayoutFromRes*. Ces données peuvent également être conservées sous la forme d'une constante dont le contenu codé en *Base64* sera inséré dans le script client. Pour faciliter cette mise en œuvre, l'exemple fourni dans le pack (*BuildConstFromBin.vbs*) permet justement de créer ou de compléter un script vbs à partir d'un fichier binaire.

Le champ texte de chaque contrôle permet d'y ajouter les valeurs décrites dans la section *référence*. Deux contrôles présentent une particularité :

- **Image (ou Bitmap)** : le style *WS_TABSTOP* doit impérativement être défini ;

- **FileDialogBox** : ce contrôle n'existe pas réellement et doit être remplacé par le contrôle *CommandButton*. C'est le format du texte (*caption|type*) qui permettra à **xGUICOM** de l'identifier comme un *FileDialogBox*.

Resource Hacker, malgré ses qualités, est un programme déjà ancien qui n'est plus mis à jour par son auteur. Il ne prend pas en compte les contrôles plus récents comme *IPControl* qui est reconnu par **xGUICOM**. Le deuxième lien concerne un éditeur dénommé **Resourcer911** ou

TrueResourcer qui est un peu plus complet et qui autorise également la sauvegarde individuelle d'une ressource sous forme binaire.

5. Analyse

Il est considéré comme acquis que le lecteur maîtrise les principes de base de la programmation *Win32*.

L'interface **xGUICOM** est constituée par des boîtes de dialogue modales générées par l'API de Windows. Leur logique de fonctionnement est celle décrite dans le chapitre "Dialog Boxes" de la documentation *MSDN Library*. En conséquence, toute la partie navigation dans l'interface est gérée automatiquement par le dialogue ce qui permet d'alléger le code. Comme le dialogue modal devait pouvoir être créé à partir de données du script et qu'il devait être également possible d'initialiser simplement les contrôles, le choix de la fonction *DialogBoxIndirectParam* s'imposait.

Après son instanciation, le composant doit appeler une fois la fonction **Initialize**. Cette dernière permet d'une part, d'assurer l'instanciation du composant *DynamicWrapperX* quel que soit le contexte, selon une technique décrite dans cet article ([Lien 137](#)) et d'autre part, de définir les différentes variables globales nécessaires à son fonctionnement. Cet appel est assuré par les méthodes **CreateForm** ou **LoadLayoutFromRes**.

Les données de chaque boîte de dialogue sont conservées dans deux buffers chaîne organisés en tableau, chaque dialogue correspondant à un élément de celui-ci. Le premier, *DLGTEMPLATEEX()*, stocke les structures *DLGTEMPLATEEX* et *DLGITEMTEMPLATEEX* définies par *Microsoft* et attendues par la fonction API *DialogBoxIndirectParam* - pour plus de détails, voir *MSDN Library* - et le second *aDataDlg()* conserve les données d'initialisation de certains contrôles qui seront traitées lors du premier appel de la **DialogProc** (cf. infra). La taille de ces buffers a été définie arbitrairement à **8192** octets et devra être éventuellement augmentée si les masques comportent de nombreux contrôles.

Les données issues des ressources binaires chargées par **LoadLayoutFromRes** sont sauvegardées dans le buffer par la fonction **ParseBinRes** et celles provenant d'**AddControl** par la fonction **BuildDataDlg**.

- Initialisation des contrôles

Certains contrôles peuvent recevoir une valeur initiale qu'il est intéressant de gérer lors de la définition des contrôles au lieu d'obliger le concepteur à programmer manuellement cette initialisation lors de l'évènement *Create*. Elle doit être effective aussi bien en création manuelle avec la méthode **AddControl** qu'avec la création assistée par un éditeur de ressources.

CommandButton, *OptionButton* et *CheckBox* peuvent,

outre le paramètre *title* définissant la légende, recevoir une valeur numérique au moyen du paramètre *style* que l'on définira aussi bien avec **AddControl** qu'avec un éditeur.

Label, *Frame*, *Edit*, *Edpswd* et *Memo* peuvent recevoir le paramètre chaîne *title* qui sera pris en compte sans traitement particulier.

ListBox, *Image*, *ComboBox*, *HotKey*, *IPControl* et *FileDialogBox* nécessitent un code d'initialisation.

ListBox : les données du paramètre *title* sont accessibles lors de l'appel initial de la *DialogProc* avec le message *WM_INITDIALOG*. La chaîne est découpée en items qui alimentent la liste.

Image : les données du paramètre *title* sont également disponibles. Il doit s'agir du nom complet d'un fichier contenant l'image à afficher au format bmp. La mise à jour est effectuée dans la **DialogProc**.

ComboBox, *HotKey*, *IPControl* : pour des raisons liées à l'interprétation du contenu, le paramètre *title* n'est pas conservé lors de l'appel initial. Il est donc nécessaire de sauvegarder les données dans un buffer avant d'appeler la fonction *DialogBoxIndirectParam*. Les fonctions **ParseHotKey** et **ParseIPStr** assurent l'encodage de ces données dans un format compréhensible pour ces contrôles.

FileDialogBox : il s'agit d'un pseudo-contrôle qui est traité comme un *CommandButton*. Le paramètre *title* contient les données du dialogue système. Elles seront traitées chaque fois que l'utilisateur actionne le contrôle bouton par la fonction **BuildOpenFileName**.

La fonction **DialogProc** est strictement celle décrite dans la documentation de la fonction API *DialogBoxIndirectParam*. Elle reçoit les notifications des contrôles et permet de définir les événements que l'on souhaite transmettre au script client. Son appel, lors de la création du dialogue, permet d'initialiser les contrôles qui nécessitent un code spécifique.

6. Liens

Téléchargement du pack complet : composant, exemple et fichier ressource : [Lien 138](#).

Site Resource Hacker : [Lien 139](#).

Editeur *Resourcer911* : [Lien 140](#).

7. Le script VBS

Les composants WSC ne supportent pas les caractères accentués lorsqu'ils ne sont pas sauvegardés en *Unicode*. Les commentaires sont donc volontairement dépourvus de toute accentuation.

Retrouvez directement le script complet à cette adresse : [Lien 141](#)

Retrouvez l'article d'*omen999* en ligne : [Lien 142](#)

Liens

- Lien 01 : <http://www.developpez.com/actu/23766/Java-SE-7-Oracle-invite-la-Fondation-Apache-a-reconsiderer-sa-position-honorez-vos-engagements-repond-la-Fondation/>
- Lien 02 : <http://www.developpez.com/actu/33040/Android-Oracle-demanderait-des-dommages-et-interets-superieurs-aux-revenus-generes-par-l-OS-depuis-son-lancement/>
- Lien 03 : <http://javaweb.developpez.com/actu/32175/Java-Oracle-veut-faire-evoluer-le-Java-Community-Process-vers-plus-de-transparence-de-reactivite-et-d-agilite/>
- Lien 04 : <http://jcp.org/en/jsr/results?id=5207>
- Lien 05 : <http://www.developpez.net/forums/d1001119/java/general-java/specifications-java-se-7-8-adoptees-jcp-jdk-7-atteint-stade-feature-complete/>
- Lien 06 : <http://www.developpez.com/actu/23364/Oracle-prepare-une-version-Premium-payante-de-la-Java-Virtual-Machine-elle-sera-issue-de-la-fusion-de-JRockit-et-de-HotSpot/>
- Lien 07 : <http://www.oracle.com/technetwork/middleware/jrockit/downloads/index.html>
- Lien 08 : http://blogs.oracle.com/henrik/entry/jrockit_is_now_free_and
- Lien 09 : <http://www.developpez.net/forums/d1084985/java/general-java/java-machine-virtuelle-jrockit-doracle-devient-gratuite/>
- Lien 10 : <http://www.scala-lang.org/node/9483>
- Lien 11 : <http://www.scala-ide.org/2011/05/scala-ide-beta-4-available/>
- Lien 12 : <http://typesafe.com/>
- Lien 13 : <http://akka.io/>
- Lien 14 : <http://www.scala-ide.org/>
- Lien 15 : <http://www.developpez.net/forums/d1082106/java/general-java/langage/scala/sortie-scala-2-9-0-final/>
- Lien 16 : <http://khayyam.developpez.com/articles/web/gwt/bundle/>
- Lien 17 : <http://developer.android.com/sdk/index.html>
- Lien 18 : <http://www.eclipse.org/downloads/>
- Lien 19 : <http://java.sun.com/javase/downloads/index.jsp>
- Lien 20 : <https://dl-ssl.google.com/android/eclipse/>
- Lien 21 : <http://developer.android.com/guide/topics/intents/intents-filters.html>
- Lien 22 : <http://developer.android.com/reference/android/view/ViewGroup.html>
- Lien 23 : <http://developer.android.com/reference/android/app/Activity.html>
- Lien 24 : <http://nbenbourahla.developpez.com/tutoriels/android/introduction-programmation-android/>
- Lien 25 : <http://nbenbourahla.developpez.com/tutoriels/android/comprendre-corriger-erreurs-application/>
- Lien 26 : <http://msdn.microsoft.com/fr-fr/asp.net/ff796201>
- Lien 27 : <http://weblogs.asp.net/scottgu/>
- Lien 28 : <http://nicolasesprit.developpez.com/tutoriels/dotnet/nouveautes-asp-net-mvc-3/>
- Lien 29 : <http://rdonfack.developpez.com/tutoriels/web/webmatrix-decouverte-et-prise-main-outil-developpement-web-gratuit-tout/>
- Lien 30 : <http://braincracking.org/?p=597>
- Lien 31 : <http://braincracking.org/>
- Lien 32 : <http://braincracking.org/2011/02/19/etat-des-lieux-de-l%E2%80%98accessibilite-de-html5/>
- Lien 33 : <http://www.w3.org/TR/html5/content-models.html#annotations-for-assistive-technology-products-aria>
- Lien 34 : <https://www.readability.com/>
- Lien 35 : <http://www.html5accessibility.com/>
- Lien 36 : <http://code.google.com/p/html5shim/>
- Lien 37 : <http://developer.yahoo.com/blogs/ydn/posts/2010/10/how-many-users-have-javascript-disabled/>
- Lien 38 : <http://webaim.org/projects/screenreadersurvey/>
- Lien 39 : <http://dev.opera.com/articles/view/more-accessible-html5-video-player/#wai-aria>
- Lien 40 : <http://praegnanz.de/html5video/>
- Lien 41 : <http://www.w3.org/TR/html5/sections.html#outlines>
- Lien 42 : <http://code.google.com/p/h5o/>
- Lien 43 : http://www.freedomscientific.com/fs_products/software_jaws.asp
- Lien 44 : <http://www.gwmicro.com/Window-Eyes/>
- Lien 45 : <http://www.nvda-project.org/>
- Lien 46 : <https://addons.mozilla.org/fr/firefox/addon/juicy-studio-accessibility-too/>
- Lien 47 : <http://jpvincent.developpez.com/tutoriels/xhtml/etat-lieux-accessibilite-html5/>
- Lien 48 : <http://www.sohtanaka.com/web-design/css-ordered-list-enhancement-tutorial/>
- Lien 49 : <http://sohtanaka.developpez.com/tutoriels/css/creer-listes-ordonnees-atrayantes-css/fichiers>
- Lien 50 : <http://jquery.com/>
- Lien 51 : <http://sohtanaka.developpez.com/tutoriels/css/creer-listes-ordonnees-atrayantes-css/>
- Lien 52 : <http://www.ibm.com/developerworks/xml/library/j-sparql/>
- Lien 53 : <http://www.w3.org/TR/rdf-sparql-query/>
- Lien 54 : <http://www.w3.org/TR/rdf-sparql-protocol/>
- Lien 55 : <http://www.w3.org/2001/sw/DataAccess/>
- Lien 56 : <http://www.developpez.net/forums/redirect-to/?redirect=http%3A%2F%2Fjena.sourceforge.net%2FARQ%2FTutorial%2F>
- Lien 57 : <http://www.ilrt.bris.ac.uk/discovery/2004/01/turtle/>
- Lien 58 : <http://www.w3.org/2000/10/swap/Primer>
- Lien 59 : ftp://ftp-developpez.com/web-semantique/tutoriels/jena/arq/introduction-sparql/fichiers/vc-db-1_rdf
- Lien 60 : <http://www.ietf.org/rfc/rfc2426.txt>
- Lien 61 : <http://www.w3.org/TR/vcard-rdf/>
- Lien 62 : <ftp://ftp-developpez.com/web-semantique/tutoriels/jena/arq/introduction-sparql/fichiers/q1.rq>
- Lien 63 : <http://jena.sourceforge.net/ARQ/cmds.html>
- Lien 64 : <http://www.cygwin.com/>
- Lien 65 : <ftp://ftp-developpez.com/web-semantique/tutoriels/jena/arq/introduction-sparql/fichiers/q-bp1.rq>
- Lien 66 : <ftp://ftp-developpez.com/web-semantique/tutoriels/jena/arq/introduction-sparql/fichiers/q-bp2.rq>
- Lien 67 : <ftp://ftp-developpez.com/web-semantique/tutoriels/jena/arq/introduction-sparql/fichiers/q-bp3.rq>
- Lien 68 : <ftp://ftp-developpez.com/web-semantique/tutoriels/jena/arq/introduction-sparql/fichiers/q-bp4.rq>
- Lien 69 : <ftp://ftp-developpez.com/web-semantique/tutoriels/jena/arq/introduction-sparql/fichiers/q-f1.rq>
- Lien 70 : <http://www.w3.org/TR/xpath-functions/#regex-syntax>
- Lien 71 : <http://jakarta.apache.org/oro/>
- Lien 72 : ftp://ftp-developpez.com/web-semantique/tutoriels/jena/arq/introduction-sparql/fichiers/vc-db-2_rdf
- Lien 73 : <ftp://ftp-developpez.com/web-semantique/tutoriels/jena/arq/introduction-sparql/fichiers/q-f2.rq>

Lien 74 : ftp://ftp-developpez.com/web-semantique/tutoriels/jena/arc/introduction-sparql/fichiers/q-opt1_rq
Lien 75 : ftp://ftp-developpez.com/web-semantique/tutoriels/jena/arc/introduction-sparql/fichiers/vc-db-2_rdf
Lien 76 : ftp://ftp-developpez.com/web-semantique/tutoriels/jena/arc/introduction-sparql/fichiers/q-opt2_rq
Lien 77 : ftp://ftp-developpez.com/web-semantique/tutoriels/jena/arc/introduction-sparql/fichiers/q-opt3_rq
Lien 78 : ftp://ftp-developpez.com/web-semantique/tutoriels/jena/arc/introduction-sparql/fichiers/q-opt4_rq
Lien 79 : <ftp://ftp-developpez.com/web-semantique/tutoriels/jena/arc/introduction-sparql/fichiers/vc-db-3.ttl>
Lien 80 : ftp://ftp-developpez.com/web-semantique/tutoriels/jena/arc/introduction-sparql/fichiers/q-union1_rq
Lien 81 : ftp://ftp-developpez.com/web-semantique/tutoriels/jena/arc/introduction-sparql/fichiers/q-union1.alt_rq
Lien 82 : ftp://ftp-developpez.com/web-semantique/tutoriels/jena/arc/introduction-sparql/fichiers/q-union2_rq
Lien 83 : ftp://ftp-developpez.com/web-semantique/tutoriels/jena/arc/introduction-sparql/fichiers/q-union3_rq
Lien 84 : ftp://ftp-developpez.com/web-semantique/tutoriels/jena/arc/introduction-sparql/fichiers/basic_patterns.html
Lien 85 : <ftp://ftp-developpez.com/web-semantique/tutoriels/jena/arc/introduction-sparql/fichiers/optionals.html>
Lien 86 : <ftp://ftp-developpez.com/web-semantique/tutoriels/jena/arc/introduction-sparql/fichiers/union.html>
Lien 87 : <ftp://ftp-developpez.com/web-semantique/tutoriels/jena/arc/introduction-sparql/fichiers/ds-dft.ttl>
Lien 88 : <ftp://ftp-developpez.com/web-semantique/tutoriels/jena/arc/introduction-sparql/fichiers/ds-ng-1.ttl>
Lien 89 : <ftp://ftp-developpez.com/web-semantique/tutoriels/jena/arc/introduction-sparql/fichiers/ds-ng-2.ttl>
Lien 90 : ftp://ftp-developpez.com/web-semantique/tutoriels/jena/arc/introduction-sparql/fichiers/q-ds-1_rq
Lien 91 : ftp://ftp-developpez.com/web-semantique/tutoriels/jena/arc/introduction-sparql/fichiers/q-ds-2_rq
Lien 92 : ftp://ftp-developpez.com/web-semantique/tutoriels/jena/arc/introduction-sparql/fichiers/q-ds-3_rq
Lien 93 : ftp://ftp-developpez.com/web-semantique/tutoriels/jena/arc/introduction-sparql/fichiers/q-ds-4_rq
Lien 94 : ftp://ftp-developpez.com/web-semantique/tutoriels/jena/arc/introduction-sparql/fichiers/q-ds-5_rq
Lien 95 : <http://web-semantique.developpez.com/tutoriels/jena/arc/introduction-sparql/>
Lien 96 : <http://trac.enlightenment.org/e/wiki/EFLxx>
Lien 97 : <http://trac.enlightenment.org/e/wiki/Python>
Lien 98 : <http://www.enlightenment.org/?p=about/sponsors>
Lien 99 : <http://docs.enlightenment.org/auto/eina/>
Lien 100 : http://svn.enlightenment.org/svn/e/trunk/EXAMPLES/eet_connection/
Lien 101 : <http://docs.enlightenment.org/auto/eet/>
Lien 102 : <http://docs.enlightenment.org/auto/evas/>
Lien 103 : <http://docs.enlightenment.org/auto/ecore/>
Lien 104 : <http://docs.enlightenment.org/auto/edje/>
Lien 105 : <http://trac.enlightenment.org/e/wiki/Elementary>
Lien 106 : <http://packages.enlightenment.org/windows/Efl-1.0.0.exe>
Lien 107 : <http://packages.enlightenment.org/windows/#filelist>
Lien 108 : http://trac.enlightenment.org/e/wiki/Installation_MacOSX
Lien 109 : http://omicron.homeip.net/projects/easy_e17/easy_e17.sh
Lien 110 : <http://dev.enlightenment.fr/%7Ecaptainigloo/>
Lien 111 : <http://trac.enlightenment.org/e/wiki/Installation>
Lien 112 : <http://docs.enlightenment.org/auto/edje/edcref.html>
Lien 113 : <http://trac.enlightenment.org/e/wiki/EdjeProgram>
Lien 114 : <http://louis-du-verdier.developpez.com/efl/debuter/>
Lien 115 : http://www.forum.nokia.com/info/sw.nokia.com/id/da8df288-e615-443d-be5c-00c8a72435f8/Ot_SDK.html
Lien 116 : <http://get.qt.nokia.com/qt/add-ons/qt-mobility-opensource-src-1.1.3.zip>
Lien 117 : <http://www.developpez.net/forums/d1028859/c-cpp/bibliotheques/qt/qt-sdk-1-1-disponible-beta/>
Lien 118 : <http://sourceforge.net/projects/qwt/>
Lien 118 : <http://www.developpez.net/forums/d1071111/c-cpp/bibliotheques/qt/outils/bibliotheques/qwt/sortie-qwt-6-0-0-a/>
Lien 119 : <http://qt.gitorious.org/qt>
Lien 120 : <http://blog.developpez.com/dourouc05/p9961/qt/qt-5/quelques-pensees-sur-qt-5-queelles-seront/>
Lien 121 : <http://www.riverbankcomputing.co.uk/static/Docs/PyQt4/html/qt.html#ImageConversionFlag-enum>
Lien 122 : <http://www.riverbankcomputing.co.uk/static/Docs/PyQt4/html/classes.html>
Lien 123 : <http://www.pyside.org/docs/pyside/>
Lien 124 : <http://www.developpez.net/forums/d1073725/autres-langages/python-zope/gui/pyside-pyqt/utilisation-dimages-pyqt/>
Lien 125 : <http://vincent-vande-vyvre.developpez.com/tutoriels/pyqt/manipulation-images/>
Lien 126 : <http://www.angusj.com/resourcehacker/>
Lien 127 : <http://home.hccnet.nl/p.vd.klugt/>
Lien 128 : <http://www.toptensoftware.com/quickprompts/>
Lien 129 : <http://www.veretennikov.org/Default.aspx?f=WSO%2FDefault.aspx>
Lien 130 : <http://omen999.developpez.com/tutoriels/vbs/RegFreeSxS/>
Lien 131 : <http://omen999.developpez.com/tutoriels/vbs/deployBin/>
Lien 132 : http://omen999.developpez.com/tutoriels/vbs/xGUICOM/?page=page_1#s3
Lien 133 : http://omen999.developpez.com/tutoriels/vbs/xGUICOM/?page=page_1#s4
Lien 134 : http://omen999.developpez.com/tutoriels/vbs/xGUICOM/?page=page_1#s5
Lien 135 : http://omen999.developpez.com/tutoriels/vbs/xGUICOM/?page=page_2#s8
Lien 136 : http://omen999.developpez.com/tutoriels/vbs/xGUICOM/?page=page_1#s6
Lien 137 : <http://omen999.developpez.com/tutoriels/vbs/deployBin/>
Lien 138 : <ftp://ftp-developpez.com/omen999/tutoriels/vbs/xGUICOM/fichiers/xGuiCom10.zip>
Lien 139 : <http://www.angusj.com/resourcehacker/>
Lien 140 : <http://www.soft-universe.com/file/Resourcer911.2009-10-23.zip>
Lien 141 : http://omen999.developpez.com/tutoriels/vbs/xGUICOM/?page=page_2
Lien 142 : <http://omen999.developpez.com/tutoriels/vbs/xGUICOM/>