# Developpez

## Le Mag

Édition de Octobre - Novembre 2011.
Numéro 36.
Magazine en ligne gratuit.

Réalisation : Alexandre Pottiez
Rédaction : la rédaction de Developpez
Contact : magazine@redaction-developpez.com

## Sommaire



## Éditorial

Le magazine de Developpez.com est de nouveau au rendez-vous pour vous dévoiler le meilleur des technologies informatiques pour votre plus grand plaisir.

Nos meilleurs articles, critiques de livres, questions/réponses, news sont à découvrir ou redécouvrir dans ce magazine.

Profitez-en bien !

La rédaction

## Article Web sémantique


### L'indexation des données dans le monde du Web sémantique

Le Web des données n'est pas très différent du Web des documents : pour que quelqu'un vienne voir quelles informations sont disponibles, il faut que cette personne puisse les trouver.

par **Thibaut Cuvelier**
Page 24

## Article DotNet


### Windows Phone 7 Mango : découvrez les nouvelles tâches

Découvrez en détail, accompagnée d'exemples de codes et de captures d'écran, les nouvelles tâches apportées par Windows Phone 7 Mango.

par **Loïc Rebours**
Page 45

## 3T : les Tests en Trois Temps

Le TDD, la fameuse méthode de Développement Guidé par les Tests est devenue incontournable. Toutefois, elle n'est pas si simple à comprendre et à mettre en œuvre. Ce mini-article propose, comme version allégée du TDD, la "3T" qui, bien qu'incomplète, devrait suffire à la plupart des équipes.

### 1. Introduction

Aujourd'hui, tous les acteurs du développement sont d'accord sur l'importance des tests. La réalisation d'une application passe nécessairement par des phases de tests. Une application non soumise aux tests est une application qui par nature sera beaucoup plus instable et donc aura toutes les chances d'être vouée à l'échec, l'abandon, ou au délaissement. Pourtant, encore trop souvent, les projets font l'impasse, espérant gagner un peu de temps. Les tests sont accusés, bien à tort, de consommer trop de charges, d'être difficiles à maintenir, etc. Or une stratégie dans laquelle les tests sont au cœur des développements peut faire gagner non seulement en fiabilité ou en crédibilité mais aussi en temps.

Une des méthodes à la mode est la fameuse TDD (Test Driven Development) ou "Développement Guidé par les Tests" qui ne sera pas expliquée ici. À la place je propose "3T", pour "Test en Trois Temps". Cette solution personnelle s'inspire des TDD. J'en propose une version allégée qui convient pour la plupart des besoins simples et, surtout, qui ne nécessite que très peu de temps pour la mettre en œuvre.

Pour découvrir les TDD, je conseille le très bon tutoriel de Bruno Irsier sur developpez.com : Lien 01.

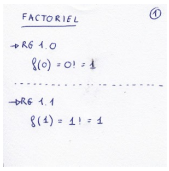
### 2. Le cahier des charges

À défaut de spécifications claires, bien rédigées, avec des règles métier correctement identifiées, il n'est pas rare qu'on ne dispose que de simples post-it. Illustrons donc la suite de cet article à l'aide de post-it.

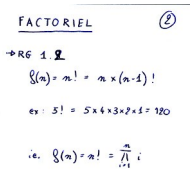
Dans la suite, et à titre d'illustration pour cet article, disons que le client demande le développement de la fonction mathématique "Factoriel" dans la couche "Service" du projet. Il fournit la série de post-it suivante :

**Regle #RG1.0**
f(n) = 1 si n == 0
ie. 0! = 1

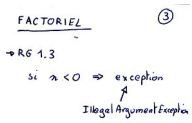
**Regle #RG1.1**
f(n) = 1 si n == 1
ie. 1! = 1




**Regle #RG1.2**
f(n) = n! = n * (n – 1)!
ex. 5! = 5 * 4 * 3 * 2 * 1 = 120




**Regle #RG1.3**
f(n) => Erreur (IllegalArgumentException) si n < 0




Si, en tant que développeur, on découvre une incohérence dans le cahier des charges, on doit en avertir le client. Il ne faut pas prendre l'initiative de modifier la demande sans l'accord du client. Dans un cas plus complexe, il se peut qu'on ait simplement mal compris les spécifications.

### 3. Mise en place

Il faut avoir un projet sur lequel travailler. Pour illustrer cet article, on crée donc un miniprojet, avec Maven, nommé "my-calculette" et dans lequel on programmera des fonctions mathématiques simples en guise d'exemple.

La doc d'installation de Maven est disponible à l'adresse suivante : Lien 02.

Sur le disque, on ajoute le dossier "my-calculette" (à l'emplacement que l'on souhaite) puis on y crée le fichier "pom.xml" avec le contenu suivant :

pom.xml
```
<project
xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
  xsi:schemaLocation="http://maven.apache.org/POM
/4.0.0 http://maven.apache.org/xsd/maven-
4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.thi</groupId>
  <artifactId>calculette</artifactId>
```

```
  <version>1.0-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>calculette</name>
  <url>http://maven.apache.org</url>

  <properties>
    <project.build.sourceEncoding>UTF-
8</project.build.sourceEncoding>
  </properties>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

Ensuite, en ligne de commande, dans le répertoire "my-calculette", on fabrique la structure du projet Eclipse à l'aide de la commande Maven suivante.

Création du projet
```
mvn clean install eclipse:eclipse
```

Si on vient tout juste d'installer Maven, celui-ci va commencer par rapatrier un nombre assez important de bibliothèques, dont JUnit, nécessaires à son fonctionnement. Donc, il est important d'avoir une connexion Internet active à ce moment-là.

La commande crée en particulier le dossier "target" et les fichiers Eclipse ".classpath" et ".project". Il ne reste plus qu'à importer le projet dans Eclipse (via le menu "File>Import>General>Existing Projects into Workspace"). Dans le projet importé, on crée un répertoire source, via un clic droit sur le projet, puis "New>Source Folder". Maintenant on peut travailler.

Eclipse et Maven ne sont utilisés ici qu'à titre pratique. Ils n'ont rien à voir avec "3T".

## 4. 3T en action

Maintenant qu'on a mis en place un projet Eclipse (my-calculette) et qu'on possède un cahier des charges, on peut s'intéresser à "3T", en trois étapes :

1. écriture des interfaces ;
2. rédaction des tests ;
3. développement assisté par les tests.

### 4.1. Interface

Puisqu'il est demandé de programmer le calcul de la fonction Factoriel dans la couche Service, on commence par déclarer cette méthode, de manière classique (et propre), dans l'interface dédiée.

L'interface
```
public interface CalculetteService {

    /**
     * Calcule le factoriel de n. <br/>
     *
     * ex. f(5) = 5! = 5 x 4 x 3 x 2 x 1 = 120.
     *
     * @param n
     * @return le factoriel de n.
     * @throws IllegalArgumentException
     *             Si la param n est négatif.
     */
    int factoriel(int n);

    ...
}
```

### 4.2. Tests

Avant même de coder l'implémentation du calcul, on écrit la classe de test. Cette seconde étape illustre ce qu'est le TDD, en créant une rupture avec les processus classiques de développement. On change ainsi d'approche, en mettant en avant le résultat à obtenir, plutôt que de commencer à s'y intéresser une fois les développements terminés ou bien avancés. Dans 3T cette seconde étape est relativement simple. En effet, il suffit de "recopier" les spécifications.

Afin d'assurer une bonne lisibilité et une bonne maintenabilité des tests, il y a deux principes qu'il convient de suivre :

1. renseigner la Javadoc de chaque méthode de test avec le résultat attendu ;
2. avoir une structure homogène du corps de chaque méthode de test, afin de faire apparaître les paramètres, les résultats attendus et les méthodes testées.

Voici un exemple de méthode de test :

Avec tous les tests
```
/**
 * Test de la regle RG1.0 <br/>
 * f(0) = 0 <br/>
 * PARAM n = 0 <br/>
 * RESULT = 1
 */
@Test
public void testCalcRG1_0() {
        // Param
        final int n = 0;

        // Resultat attendu
        final int result = 1;

        // Appel
        final int resultatCalcule =
calc.factoriel(n);

        // Test
        assertEquals(result, resultatCalcule);
}
```

Pour l'écriture d'une méthode de test, on se souvient de la règle des 3 A(s) : Arrange (préparation des objets et mocks pour le test) puis Act (appel de la méthode à tester) et enfin Assert (lignes d'assertions). Ici le premier "A" est composé des blocs "param" et "résultat attendu".

On reproduit ce schéma pour toutes les règles du cahier des charges. Comme les appels et les tests sont globalement tous les mêmes, on peut en profiter pour les

factoriser.

Format

```
public class CalculetteServiceTest {
	/**
	 * Le service de calcul
	 */
	protected CalculetteService calc;

	/**
	 * Test de la regle RG1.0 <br/>
	 * f(0) = 0 <br/>
	 * PARAM n = 0 <br/>
	 * RESULT = 1
	 */
	@Test
	public void testCalcRG1_0() {
		// Param
		final int n = 0;

		// Resultat attendu
		final int result = 1;

		// Appel et test
		doTestRG1_x(n, result);
	}

	/**
	 * Test de la regle RG1.1 <br/>
	 * f(1) = 0 <br/>
	 * PARAM n = 1 <br/>
	 * RESULT = 1
	 */
	@Test
	public void testCalcRG1_1() {
		// Param
		final int n = 1;

		// Resultat attendu
		final int result = 1;

		// Appel et test
		doTestRG1_x(n, result);
	}

	/**
	 * Test de la regle RG1.2 <br/>
	 * f(5) = 120 <br/>
	 * PARAM n = 5 <br/>
	 * RESULT = 120
	 */
	@Test
	public void testCalcRG1_2() {
		// Param
		final int n = 5;

		// Resultat attendu
		final int result = 120;

		// Appel et test
		doTestRG1_x(n, result);
	}

	/**
	 * Test de la regle RG1.3 <br/>
	 * f(-3) =- ERROR <br/>
	 * PARAM n = -3 <br/>
	 * RESULT = IllegalArgumentException
	 */
	@Test(expected =
IllegalArgumentException.class)
	public void testCalcRG1_3() {
		// Param
		final int n = -3;

		// Resultat attendu
		final int result = 1;

		// Appel et test
		doTestRG1_x(n, result);
	}

	/**
	 * Fait les appels et les tests pour les
regles RG 1.x <br/>
	 * PARAM n <br/>
	 * PARAM result
	 */
	private void doTestRG1_x(final int n,
final int result) {
		// Appel
		final int resultatCalcule =
calc.factoriel(n);

		// Test
		assertEquals(result,
resultatCalcule);
	}

}


public class RecursiveCalculetteServiceTest
extends CalculetteServiceTest {

	/**
	 * Set up
	 */
	@Before
	public void doBefore() {
		// calc = ...
	}
}
```

La classe *CalculetteServiceTest* doit être "abstraite" et permettre de faire des tests génériques quelle que soit l'implémentation du service.

Dans le cas de la règle RG1.2, on peut écrire plusieurs tests avec différentes valeurs de n, prises au hasard. Quand on souhaite tester plusieurs valeurs, on peut soit regrouper les tests dans une seule méthode soit écrire une méthode pour chaque valeur. C'est ce second choix que je propose dans le cadre de 3T. On aura ainsi un code ressemblant au suivant.

Test de plusieurs valeurs

```
	/**
	 * Test de la regle RG1.2-A <br/>
	 * f(5) = 120 <br/>
	 * PARAM n = 5 <br/>
	 * RESULT = 120
	 */
	@Test
	public void testCalcRG1_2_120() {
		// Param
		final int n = 5;
```

```
                // Resultat attendu
                final int result = 120;

                // Appel et test
                doTestRG1_x(n, result);
        }

        /**
         * Test de la regle RG1.2-24 <br/>
         * f(4) = 24 <br/>
         * PARAM n = 4 <br/>
         * RESULT = 24
         */
        @Test
        public void testCalcRG1_2_24() {
                // Param
                final int n = 4;

                // Resultat attendu
                final int result = 24;

                // Appel et test
                doTestRG1_x(n, result);
        }

        /**
         * Test de la regle RG1.2-720 <br/>
         * f(6) = 720 <br/>
         * PARAM n = 6 <br/>
         * RESULT = 720
         */
        @Test
        public void testCalcRG1_2_C() {
                // Param
                final int n = 6;

                // Resultat attendu
                final int result = 720;

                // Appel et test
                doTestRG1_x(n, result);
        }

        ...
```

On lance la série de tests (bouton droit sur le nom de la classe, puis menu *Run As > JUnit test*) et on constate que tous les tests sont rouges. C'est normal puisque la méthode *"doBefore()"* ne définit pas encore d'implémentation. On n'obtient que des NullPointerException.

On va donc maintenant créer une première implémentation, qui ne fait que compiler.

Implémentation minimale

```
public class RecursiveCalculetteService
implements CalculetteService {

        @Override
        public int factoriel(int n) {
                throw new
UnsupportedOperationException("Cette méthode n'a
pas encore été écrite.");
        }
}
```

Puis on finit la méthode *"doBefore()"* en utilisant cette implémentation.

Set up

```
public class RecursiveCalculetteServiceTest {

        /**
         * Set up
         */
        @Before
        public void doBefore() {
                calc = new
RecursiveCalculetteService();
        }

        ...
}
```

On relance les tests, qui restent rouges. Toutefois, la classe de tests est désormais finie et on peut la mettre de côté.

### 4.3. Développement

Dans cette troisième et dernière étape, il s'agit de programmer la fonctionnalité demandée à proprement parler. Il faut procéder par itération, en s'aidant de la classe de tests qu'on a laissée ouverte dans un coin d'Eclipse. L'idée est de n'écrire que du code pour faire passer les tests au vert, l'un après l'autre, un seul à la fois. Le mieux est de le faire dans l'ordre des tests quand c'est possible.

On commence donc par la première règle : RG1.0

RG1.0

```
@Override
public int factoriel(int n) {

        // RG1.0 f(0) = 1
        if(n == 0) {
                return 1;
        }

        throw new
UnsupportedOperationException("Cette partie de la
méthode n'a pas encore été écrite.");
}
```

On relance donc la série de tests et on constate que le test *"testCalcRG1_0"* devient vert, tandis que les trois autres restent rouges.

On continue avec la règle suivante : RG1.1.

RG1.1

```
@Override
public int factoriel(int n) {

        // RG1.0 f(0) = 1
        // RG1.1 f(1) = 1
        if(n == 0 || n == 1) {
                return 1;
        }

        throw new
UnsupportedOperationException("Cette partie de la
méthode n'a pas encore été écrite.");
}
```

On relance donc la série de tests et on constate que le test

"testCalcRG1_1" devient vert, que le test *"testCalcRG1_0"* reste vert, et que les 2 autres restent rouges.

On continue avec la règle suivante : RG1.2.

RG1.2
```
@Override
public int factoriel(int n) {

        // RG1.0 f(0) = 1
        // RG1.1 f(1) = 1
        if (n == 0 || n == 1) {
                return 1;
        }

        // RG1.2 f(n) = n * f(n - 1)
        return n * factoriel(n - 1);
}
```

On relance donc la série de tests et on constate que le test *"testCalcRG1_2"* devient vert, que les tests *"testCalcRG1_1"* et *"testCalcRG1_0"* restent verts, et que le dernier test reste rouge.

On continue avec la dernière règle : RG1.3.

RG1.3
```
@Override
public int factoriel(int n) {

        // RG1.3 Si n < 0 => ERREUR
        if(n < 0) {
                throw new
IllegalArgumentException("Le param n ne peut pas
être négatif.");
        }

        // RG1.0 f(0) = 1
        // RG1.1 f(1) = 1
        if (n == 0 || n == 1) {
                return 1;
        }

        // RG1.2 f(n) = n * f(n - 1)
        return n * factoriel(n - 1);
}
```

On relance donc la série de tests et on constate que tous les tests sont désormais verts. À ce stade, on est en droit de penser que (sauf hasard malheureux) le développement de la fonctionnalité est terminé.

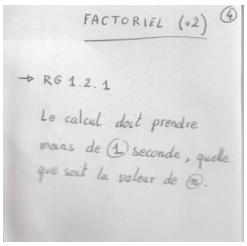
Pour finir on lance l'ensemble des tests de l'application, et non pas seulement de la classe (sous réserve que l'application contienne d'autres tests, ce qui devrait être le cas si le calcul du Factoriel n'est pas la première fonctionnalité de la calculette, cf. Chapitre "V Les évolutions"), pour vérifier que le développement n'a pas introduit de régression. Si tous les tests sont verts, c'est qu'il n'y a rien à faire et on peut raisonnablement se dire que le développement est terminé. Par contre, si un ancien test devient rouge, c'est qu'on a créé une régression, qu'il faut corriger.

Si on dispose d'un serveur d'intégration, on peut se contenter de ne lancer les tests que sur un sous-ensemble des tests de l'application pour vérifier la non-régression. En effet, le serveur d'intégration se chargera alors de lancer l'intégralité des tests. L'idée est qu'il peut y avoir des milliers de tests dans l'application et les lancer peut prendre beaucoup de temps.

## 5. Les évolutions, bogues et mises à jour

C'est inévitable, à un moment ou un autre, le client va commander des évolutions de l'application. Voici un exemple d'évolution évidemment présenté sous forme de post-it.

**Regle #RG1.2.1**
Le calcul prend moins de 1 seconde.




À partir de là, on repart directement sur "3T".

Pour l'étape 1 (Interface) il n'y a rien à faire dans l'exemple.

Pour l'étape 2, il faut ajouter un nouveau test.

RG1.2.1
```
/**
 * Test de la regle RG1.2.1 <br/>
 * f(5) = 120 <br/>
 * PARAM n = 5 <br/>
 * RESULT = 120
 * DUREE = 1 seconde max
 */
@Test(timeout=1000)
public void testCalcRG1_2_1() {
        testCalcRG1_2();
}
```

Au départ, on s'intéresse uniquement au(x) nouveau(x) test(s) qu'on doit faire passer au vert. S'ils sont verts, c'est qu'il n'y a rien à faire. Ici on ajoute une contrainte de performance donc c'est possible, mais pour une nouvelle fonctionnalité, ça n'arrive jamais. Si un nouveau test est rouge, on passe à la troisième étape.

Pour l'étape 3 (nouveau dév), on programme la nouvelle fonctionnalité et on relance les nouveaux tests. Quand les nouveaux tests sont verts, on peut considérer que les nouveaux développements (pris seuls) sont OK. Une fois que les nouveaux tests sont verts, on relance l'ensemble des tests pour vérifier la non-régression.

Quand un bogue est découvert, il donne généralement lieu à la création d'un ticket dans le système de gestion des anomalies (Mantis, Jira, etc.) Le ticket est alors référencé par un numéro unique et peut faire office de nouveau post-it. La correction d'un bogue peut se dérouler comme le développement d'une nouvelle fonctionnalité, c'est-à-dire comme indiqué plus haut.

Quand le cahier des charges évolue, et plus particulièrement quand une règle (déjà développée) change, je conseille de ne pas versionner ladite règle mais d'en créer une nouvelle (avec un nouveau numéro). Si la nouvelle règle est incompatible avec l'ancienne, on supprime simplement l'ancienne règle, devenue fausse.

## 6. Suivi

Un point très intéressant de "3T" est le suivi (reporting) qu'on peut/doit mettre en place. En effet "3T" propose de recopier les spécifications pour écrire les tests. Or les spécifications définissent très précisément le contenu de l'application. Par association, lorsque N % des tests passent au vert, cela "signifie" que N % du cahier des charges est satisfait, c'est-à-dire que l'application est développée à hauteur de N %.

Des logiciels, liés au serveur d'intégration, permettent de suivre l'avancement des développements et de donner de la visibilité au chef de projet. En effet, le serveur d'intégration récupère la dernière version du code de manière régulière (par exemple tous les soirs, ou après chaque commit), puis le compile et lance les tests. Le chef de projet peut alors consulter les résultats.

## 7. Conclusion

Sans suivre tous les principes du TDD, la "3T" permet de développer une application rapidement, avec un taux de confiance élevé et en forçant les membres de l'équipe à documenter. La "3T" permet de couvrir l'ensemble des spécifications à moindres frais et de façon relativement automatisée ; il suffit de recopier. Cela évite notamment aux développeurs de se poser mille questions, dans tous les sens, et de produire exactement ce qui est demandé, sans rien oublier ni ajouter en trop.

## 8. Annexes

### 8.1. Quelques réponses

#### 8.1.1. "3T" diverge pas mal du TDD.

Oui c'est exact et c'est assumé. Faire du Test Driven Dev, c'est se donner les moyens de réussir ses projets. Néanmoins, peu de CP (Chef de Projet) sont prêts à se lancer dans l'aventure. La mise en œuvre de la "3T" ne nécessite que très peu d'investissements. Le choix et l'écriture des cas de test est quasi systématique puisqu'il suffit de recopier les spécifications. Certains outils (non présentés dans ce document) permettent même de générer du code de test directement à partir du cahier des charges, par exemple depuis un Wiki. En outre, certains concepts des TDD, notamment en ce qui concerne la manière de faire initialement échouer des tests, ne sont pas évidents à comprendre. "3T" est alors un compromis qui semble cohérent dans de nombreux cas.

#### 8.1.2. La méthode de calcul du factoriel proposée renvoie 0 quand N vaut 50.

Oui et c'est normal. En Java, les "int" sont codés sur 32 bits. Les chiffres positifs vont de 0 à 2^31-1. Quand on calcule "50!" on dépasse la capacité des "int". D'ailleurs on dépasse la capacité max dès que N vaut 28 ou plus. On pourrait utiliser des "long", codés sur 64 bits mais ça ne ferait que reculer le problème. On pourrait aussi utiliser des BigInteger, déjà mieux adaptés. Mais l'idéal est encore d'utiliser des techniques spécifiques aux grands nombres. Toutefois ce n'est pas l'objet de ce document. La méthode de calcul du factoriel est ici proposée à titre d'illustration.

*Retrouvez l'article de Thierry Leriche-Dessirier en ligne : Lien 03*
*Retrouvez également la suite de l'article : Lien 04*

## Brevets : Samsung capitule

### Et s'engage à payer des redevances à Microsoft pour chaque dispositif Android vendu

Après HTC (Lien 05), c'est au tour d'un autre acteur majeur du marché des smartphones de venir élargir la liste des constructeurs qui payent des redevances à Microsoft pour leurs terminaux sous Android.

Microsoft vient d'annoncer un accord de licence croisé avec le constructeur Samsung pour que les deux firmes puissent bénéficier respectivement de leurs portefeuilles de brevets.

La firme de Redmond n'a pas révélé les détails sur les termes du contrat de licence, mais celui-ci prévoit le paiement des royalties pour chaque dispositif Android vendu par le constructeur coréen.

Il avait été rapporté par une source anonyme début juillet que des négociations étaient en cours entre les deux entreprises pour réduire le montant des redevances demandées par Microsoft de 15 à 10 dollars (lire ci-avant).

Les termes de l'accord ouvrent également la porte à un partenariat approfondi pour le développement de nouveaux modèles de smartphones sous Windows Phone par Samsung. Le constructeur prépare le lancement de plusieurs terminaux sous Windows Phone 7.5 Mango dont le premier modèle Omnia W est annoncé pour mi-novembre (Lien 06).

En seulement trois mois, Microsoft a signé des accords avec Acer (Lien 07), General Dynamics Itronix, Onkyo, Velocity Micro, ViewSonic (Lien 07) et Wistron.

Le seul acteur majeur sur le marché des smartphones qui ne s'est pas encore plié aux exigences de Microsoft est Motorola Mobility, désormais filiale de Google. Microsoft avait attaqué Motorola en justice pour violation de sept de ses brevets dans les téléphones Android du constructeur (Lien 08).

Cet accord permet néanmoins à Samsung de se concentrer sur les multiples procès dans différents pays contre Apple : Lien 09.

*Commentez la news d'Idelways en ligne : Lien 10*

## Google s'offre Motorola Mobility

### Pour 12.5 milliards de dollars

Une grande nouvelle vient secouer le monde IT et le sortir de sa lourde torpeur du mois d'août, une nouvelle que peu d'analystes peuvent prétendre avoir vue venir.

Google vient d'annoncer par la voix de son PDG Larry Page avoir conclu un accord d'achat de la division « Mobility » du géant historique de la téléphonie mobile Motorola et... ses dizaines de milliers de brevets.

Cet accord, qui s'élève à la modique somme de 12,5 milliards de dollars (7.7 milliards d'euros), permettra à Google d'étendre son activité dans le marché des smartphones, et de se lancer sans retenue dans le domaine du hardware.

Certes, Motorola n'est à présent pas ce qu'il a été, déficitaire même de quelques centaines de millions de dollars sur l'exercice passé, mais en trouvant un acquéreur aussi bien placé que Google dans le domaine des OS mobiles, l'entreprise pourrait retrouver son lustre d'antan.

Le fait que Motorola ait passé la totalité de ses modèles de smartphones et tablettes sous Android n'en ferait que faciliter cette transition, en théorie.

Google, qui n'a de cesse d'augmenter ses effectifs rendant sa gestion interne de plus en plus compliquée, engloutira des dizaines de milliers d'employées, entièrement étrangers à la culture de l'entreprise.
Et sur ce point, autant dire que Google n'a aucune expérience, beaucoup plus habituée en effet aux achats de petites startups innovantes, plutôt que des mastodontes historiques de la taille de Motorola.

Mais le plus sérieux problème auquel s'expose Google est d'alarmer les autres constructeurs de smartphones en favorisant sa future division Motorola sur son système d'exploitation mobile open-source Android.

Larry Page se montre sur ce point rassurant en affirmant que sa division restera séparée et traitée comme égale aux autres constructeurs de téléphones. Faute de quoi, Page est certainement conscient que Windows Phone 7 et les autres alternatives en gestation s'en sortiront les plus gagnants de cette transaction de l'année.

Sans oublier le fait que le gouvernement américain et les organismes antitrust peuvent mettre des bâtons dans les roues de Google, comme il l'avait fait pour AT&T plus tôt cette année.

Aussi insurmontables que soient les difficultés qui s'opposent à cette acquisition, le coup de maître dans cette affaire est que Google mettra la main sur 17 000 brevets et 7500 autres brevets en attente d'enregistrement.

Autant de brevets qui pourraient débarrasser Android de plusieurs poursuites de plus en plus gênantes pour son expansion, directement, ou pas : échange de bons précédés avec les détenteurs d'autres brevets tiers que la commercialisation d'appareils sous Android enfreints.

Quoi qu'il en soit, Google ne semble pas avoir conclu un achat centré sur les brevets, étant donné qu'il aurait pu s'offrir des licences pour tous les brevets en litige, et pour beaucoup moins cher.

Android et mobilité sont en passe de devenir le véritable cœur de métier de Google.

*Commentez la news de Simon Bernier en ligne : Lien 11*

## Créer une soundboard

Une soundboard consiste à proposer un panel de bouton jouant chacun un son particulier sur le thème choisi. On ne le dirait pas comme ça, mais les soundboards sont des applications très appréciées par les utilisateurs. Sur Android Market par exemple, on peut retrouver des milliers d'applications de soundboards sur des thèmes variés aussi bien sur les jeux vidéo, séries télévisées ou encore de chanteurs/chanteuses.
L'objectif de cet article reste simple : faire une application Android affichant deux boutons et jouant chacun un son lorsque l'on appuie dessus. Après avoir réalisé ceci, vous serez de vous-même capable, grâce à votre imagination, de proposer une jolie interface présentant plusieurs boutons jouant différents sons.

### 1. SoundPool vs. MediaPlayer

Tout d'abord, il faut savoir que sur Android, il existe deux classes pour jouer des sons :

- la classe SoundPool : Lien 12;
- la classe MediaPlayer : Lien 13.

Je vous conseille grandement de jeter un coup d'œil à leur documentation afin d'avoir un bref aperçu de ce que nous allons traiter dans cet article.

En examinant bien la description de chacune des classes, on remarquera vite que la classe *SoundPool* ne nous intéressera pas dans notre cas. En effet, l'objectif de ce dernier est de pouvoir charger en mémoire des sons et ainsi de pouvoir les jouer autant qu'on veut dans l'application sans devoir souffrir d'une latence de chargement et de décompression du fichier audio. Bien sûr il a d'autres avantages, mais je vous laisse les découvrir par vous-même.

À savoir que ce que nous entendons par sons, ce sont des clips, bruitages ou effets, majoritairement très courts, que l'on rencontre souvent dans les jeux (explosion, laser, porte…).

Ainsi, le *SoundPool* est pratique mais impose une taille limite pour le chargement de fichiers audio. En cas de dépassement, rapidement atteint si le son dure plus de cinq secondes, cela n'influencera en rien l'exécution du programme, mais simplement que le son en question ne sera jamais entendu par l'utilisateur. Cependant, si vous faites attention à la sortie LogCat de l'appareil, vous constaterez plusieurs messages d'erreur venant de la part de cette classe.

Pour la création de soundboards, il est très probable que l'on veuille jouer des sons qui risquent de dépasser cette taille limite. De plus, pour une application de ce type, le chargement du fichier à chaque lecture aura très peu d'impact sur l'exécution du programme. Ainsi voilà pourquoi nous utiliserons la classe *MediaPlayer* fournie par la librairie Android.

Dans le projet que nous allons réaliser, on développera une simple soundboard avec ces deux sons :

- Mario - 1up : Lien 14 ;
- Mario - Coin : Lien 15.

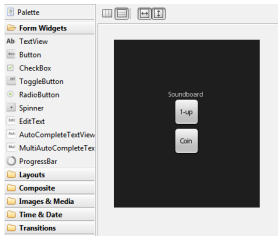
À noter que la classe *MediaPlayer* est capable de lire toute sorte de formats tels que le mp3, ogg, wav, mid… et permet donc aussi bien de jouer des sons que des musiques entières.

### 2. Création du projet

On va tout d'abord créer et préparer l'interface de l'application. Commencez par créer un simple projet Android (du nom et de l'API que vous voulez) dans votre environnement, puis allez éditer le layout principal (**res/layout/main.xml**) pour y ajouter deux boutons qu'on nommera respectivement *btn_sound_1up* et *btn_sound_coin* avec comme label (texte) *1-up* et *Coin*.

Pour les plus paresseux, voici le XML de notre layout à copier/coller dans votre **main.xml** :

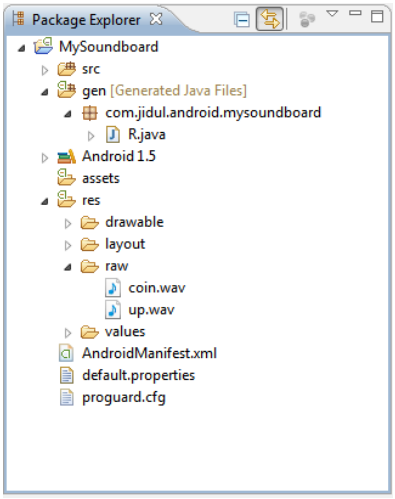
```
< ?xml version="1.0" encoding="utf-8"?>
<linearlayout
xmlns:android="http://schemas.android.com/apk/res
/android"
   android:orientation="vertical"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:gravity="center">
<textview android:layout_height="wrap_content"
android:text="Soundboard"
android:layout_width="wrap_content"/>
<button android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/btn_sound_1up" android:text="1-
up"></button>
<button android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Coin"
android:id="@+id/btn_sound_coin"></button>
</linearlayout>
```



main.xml

## 3. Ajout des sons dans le projet

Pour pouvoir jouer les sons, ces derniers devront forcément être présents quelque part. Dans notre cas, nous allons les inclure dans l'APK de l'application. Pour cela, créez un nouveau dossier dans le répertoire **res** et nommez-le **raw**. Copiez-y les deux sons fournis auparavant. Ainsi, ils vont donc être ajoutés en tant que ressources à l'application, et seront accessibles depuis la classe *R*, comme une image placée dans le dossier **drawable**.




Architecture

## 4. Lire les sons dans l'application

Nous voici maintenant dans la partie la plus intéressante : nous allons écrire le code permettant de jouer un son disponible en tant que ressource et associer un bouton à un son.

Ouvrez donc tout d'abord le code de votre *Activity* principale, et vérifiez que le layout créé auparavant a bien été associé à celle-ci :

```
setContentView(R.layout.main);
```

Ensuite, déclarons tout d'abord comme attribut à cette classe un objet de type *MediaPlayer* :

```
private MediaPlayer mPlayer = null;
```

Puis écrivons la méthode permettant de jouer la ressource passée en paramètre :

```
private void playSound(int resId) {
  if(mPlayer != null) {
    mPlayer.stop();
    mPlayer.release();
  }
  mPlayer = MediaPlayer.create(this, resId);
  mPlayer.start();
}
```

La classe *MediaPlayer* possède une méthode statique *create*, prenant comme paramètres le contexte et l'identifiant de la ressource à jouer, puis renvoyant une instance d'elle-même avec le son chargé et prêt à être joué. Il suffit alors d'appeler la méthode *start* à partir de cette dernière afin de pouvoir enfin écouter le clip.

Il n'est pas nécessaire d'appeler la méthode *prepare* de l'instance renvoyée par la méthode *create*, ce qui d'ailleurs provoquerait une exception, puisqu'elle se charge elle-même d'y faire l'appel : Lien 16.

Par la suite, il est aussi recommandé d'appeler la méthode *release* afin de libérer la ressource et donc de l'espace mémoire. Pour faire cela proprement, on surchargera aussi la méthode *onPause* de l'*Activity* pour libérer la ressource avant la fermeture de l'application :

```
@Override
public void onPause() {
  super.onPause();
  if(mPlayer != null) {
    mPlayer.stop();
    mPlayer.release();
  }
}
```

Et pour finir, appelons la méthode à chaque clic sur un bouton en lui attribuant la ressource à jouer :

```
Button btn_sound_1up = (Button)
findViewById(R.id.btn_sound_1up);
btn_sound_1up.setOnClickListener(new
OnClickListener() {

  @Override
  public void onClick(View v) {
    playSound(R.raw.up);
  }

});

Button btn_sound_coin = (Button)
findViewById(R.id.btn_sound_coin);
btn_sound_coin.setOnClickListener(new
OnClickListener() {

  @Override
  public void onClick(View v) {
    playSound(R.raw.coin);
  }

});
```

Voici le code entier de l'*Activity* :

```
package com.jidul.android.mysoundboard;

import android.app.Activity;
import android.media.MediaPlayer;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

public class MySoundboard extends Activity {
  private MediaPlayer mPlayer = null;

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

```
    Button btn_sound_1up = (Button)
findViewById(R.id.btn_sound_1up);
    btn_sound_1up.setOnClickListener(new
OnClickListener() {

      @Override
      public void onClick(View v) {
        playSound(R.raw.up);
      }

    });

    Button btn_sound_coin = (Button)
findViewById(R.id.btn_sound_coin);
    btn_sound_coin.setOnClickListener(new
OnClickListener() {

      @Override
      public void onClick(View v) {
        playSound(R.raw.coin);
      }

    });
  }

  @Override
  public void onPause() {
    super.onPause();
    if(mPlayer != null) {
      mPlayer.stop();
      mPlayer.release();
    }
  }
```

```
  private void playSound(int resId) {
    if(mPlayer != null) {
      mPlayer.stop();
      mPlayer.release();
    }
    mPlayer = MediaPlayer.create(this, resId);
    mPlayer.start();
  }
}
```

Maintenant, vous n'avez plus qu'à lancer et tester votre application dans l'émulateur ou sur votre téléphone pour vérifier que tout fonctionne correctement en appuyant sur les boutons.

Projet final

À partir de maintenant, vous êtes donc aussi capable de créer votre propre soundboard et ainsi d'enrichir à votre tour l'Android Market !

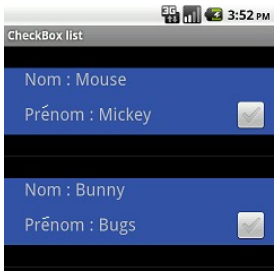
*Retrouvez l'article de Jonathan Odul en ligne : Lien 17*

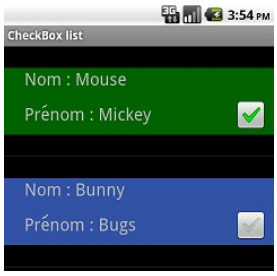
## Créer une ListView avec des contrôles CheckBox et gérer les événements sur ces derniers

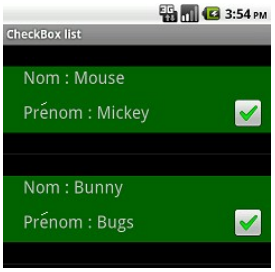
L'application que je vous propose est assez classique et simple. Il s'agit de créer une liste comportant des checkbox. Au clic d'une checkbox, ceci va faire changer la couleur de fond du bloc de celle-ci. Attention: suite à quelque bogue sur ce tutoriel, je vous informe que ce tutoriel fonctionne seulement si la hauteur de la ListView ne dépasse pas la hauteur de l'écran (il ne faut pas avoir de scroll). Une solution est de créer une pagination pour passer à la suite de la liste.

### 1. Introduction

Voici des captures d'écran de l'application :



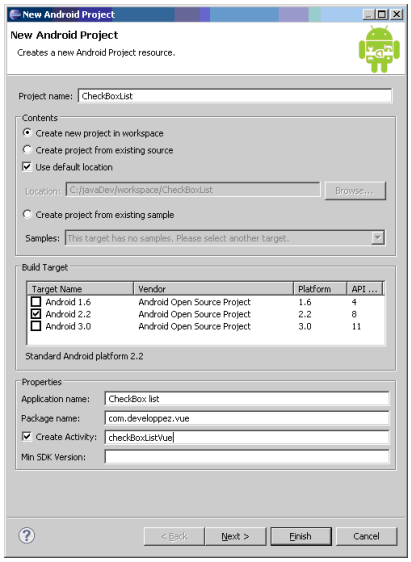




Les technologies utilisées dans ce tutoriel sont :
- listView ;
- listActivity ;
- événement sur checkbox ;
- adaptateur.

## 2. Création de l'application

Pour cela nous allons commencer par créer une application :




## 3. Les vues

Commençons par les vues.
La première est le main.xml. Elle comprend la listView (la liste). Sachant que nous allons utiliser une ListActivity par la suite, l'identifiant de la listView sera @android:id/list. Ce qui permettra à l'activité de reconnaitre automatiquement la liste.
Voici le code de la vue qui est assez simple :

main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res
/android"
   android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical">

  <ListView android:id="@android:id/list"
android:layout_width="fill_parent"
     android:layout_height="fill_parent" />

</LinearLayout>
```

La deuxième vue, nommée list_detail.xml, correspond au contenu de chaque ligne de la liste. Dans notre exemple, nous avons en tout quatre champs texte (deux libellés et deux champs modifiables) et une checkbox. Les champs texte sont placés dans des tableaux afin de pouvoir les intégrer correctement avec la checkbox. Mais là, libre à vous de placer vos champs où vous voulez !
Pour la checkbox, nous définissons la fonction qui permettra de récupérer l'événement dans l'activité comme ceci android:onClick="MyHandler". L'identifiant de la checkbox est check, celle du champ modifiable contenant le nom est nom et celui du prénom est prenom.
Pour une raison ergonomique, nous choisissons de colorer une partie du bloc. Cette partie est celle définie par l'identifiant blocCheck.
Voici le code :

list_detail.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res
/android"
   android:orientation="horizontal"
android:layout_width="fill_parent"
   android:layout_height="wrap_content">

  <!-- Tableau permettant d'afficher tout le
contenu d'un bloc -->
  <TableLayout android:id="@+id/blocCheck"
     android:background="@color/blue"
     android:layout_height="wrap_content"
     android:layout_width="fill_parent"
     android:layout_marginTop="25px"
     android:layout_marginBottom="25px">

  <!-- La première ligne affiche deux libellés
côte à côte (le nom et sa valeur) -->
    <TableRow android:layout_width="fill_parent"
       android:layout_height="wrap_content"
       android:paddingLeft="4sp"
       android:paddingBottom="4sp">
      <LinearLayout
         android:orientation="horizontal"
         android:layout_width="fill_parent"
         android:layout_height="wrap_content"
         android:paddingLeft="20sp"
         android:layout_weight="1"
        android:layout_gravity="center_vertical">

        <TextView
           android:layout_width="wrap_content"
           android:layout_height="wrap_content"
           android:textSize="20sp"
           android:text="@string/nom"
           android:layout_gravity="left"
           android:layout_marginRight="5sp" />
        <TextView android:id="@+id/nom"
           android:layout_width="wrap_content"
           android:layout_height="wrap_content"
           android:textSize="20sp"
           android:layout_gravity="left" />
      </LinearLayout>
    </TableRow>

  <!-- Cette ligne affiche les deux libellés du
prénom et la checkbox -->
    <TableRow android:layout_width="fill_parent"
       android:layout_height="wrap_content"
       android:paddingLeft="4sp"
       android:baselineAligned="true">
      <LinearLayout
         android:orientation="horizontal"
         android:layout_width="fill_parent"
         android:layout_height="wrap_content"
         android:paddingLeft="20sp"
         android:layout_weight="1"
        android:layout_gravity="center_vertical">
        <TextView
           android:layout_width="wrap_content"
           android:layout_height="wrap_content"
           android:textSize="20sp"
           android:text="@string/prenom"
           android:layout_gravity="left"
           android:layout_marginRight="5sp" />
        <TextView android:id="@+id/prenom"
           android:layout_width="wrap_content"
```

```
            android:layout_height="wrap_content"
            android:textSize="20sp"
            android:layout_gravity="left" />
      </LinearLayout>
      <CheckBox
         android:layout_height="wrap_content"
         android:id="@+id/check"
         android:layout_width="wrap_content"
         android:layout_gravity="right"
         android:layout_marginRight="10sp"
         android:onClick="MyHandler"/>
    </TableRow>
  </TableLayout>
</LinearLayout>
```

Les vues sont créées.

## 4. Les paramètres

Comme vous avez pu le remarquer, nous utilisons des variables définies dans d'autres fichiers xml afin de pouvoir internationaliser notre application si nous le souhaitons. Ainsi, nous avons un fichier string.xml comportant les différents libellés utilisés comme ceci :

string.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World,
checkBoxListVue!</string>
    <string name="app_name">CheckBox
list</string>
    <string name="nom">Nom : </string>
    <string name="prenom">Prénom : </string>
</resources>
```

De même, nous allons utiliser des couleurs. Donc afin de pouvoir les modifier facilement et aussi de pouvoir les utiliser convenablement nous créons un fichier color.xml comme ceci :

color.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="blue">#3050A0</color>
    <color name="green">#006600</color>
</resources>
```

## 5. L'activité

Écrivons maintenant l'activité.
Cette activité remplit ma liste et utilise un adaptateur redéfini par une classe écrite plus bas. Elle contient la fonction publique void MyHandler(View v) qui permet de récupérer l'événement du clic sur la checkbox (rappelez-vous, nous avons défini cette fonction dans la vue au niveau de la checkbox).
Voici le code commenté de l'activité :

checkBoxListVue.java

```
package com.developpez.vue;

import java.util.ArrayList;
import java.util.HashMap;
import android.app.ListActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.CheckBox;
import android.widget.ListView;

public class checkBoxListVue extends ListActivity
{
  private ListView list;

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

    //Récupération automatique de la liste (l'id
de cette liste est nommé obligatoirement
@android:id/list afin d'être détecté)
    list = getListView();

    // Création de la ArrayList qui nous
permettra de remplir la listView
    ArrayList<HashMap<String, String>> listItem =
new ArrayList<HashMap<String, String>>();

    // On déclare la HashMap qui contiendra les
informations pour un item
    HashMap<String, String> map;

    map = new HashMap<String, String>();
    map.put("nom", "Mouse");
    map.put("prenom", "Mickey");
    listItem.add(map);

    map = new HashMap<String, String>();
    map.put("nom", "Bunny");
    map.put("prenom", "Bugs");
    listItem.add(map);

    //Utilisation de notre adaptateur qui se
chargera de placer les valeurs de notre liste
automatiquement et d'affecter un tag à nos
checkbox

    MyListAdapter mSchedule = new
MyListAdapter(this.getBaseContext(), listItem,
R.layout.list_detail, new String[] { "nom",
"prenom" }, new int[] {
      R.id.nom, R.id.prenom });

    // On attribue à notre listView l'adaptateur
que l'on vient de créer
    list.setAdapter(mSchedule);
  }

  //Fonction appelée au clic d'une des checkbox
  public void MyHandler(View v) {
    CheckBox cb = (CheckBox) v;
    //on récupère la position à l'aide du tag
défini dans la classe MyListAdapter
    int position =
Integer.parseInt(cb.getTag().toString());

    // On récupère l'élément sur lequel on va
changer la couleur
    View o = list.getChildAt(position)
              .findViewById(R.id.blocCheck);

    //On change la couleur
    if (cb.isChecked()) {
      o.setBackgroundResource(R.color.green);
    } else {
      o.setBackgroundResource(R.color.blue);
```

```
    }
  }
}
```

## 6. L'adaptateur

Il ne reste plus qu'à définir la classe MyListAdaptater qui permettra surtout de définir un tag pour chaque checkbox. Ce tag correspond à la position du bloc (l'item) dans lequel est située la checkbox. Ceci est défini dans la fonction getView.
Voici le code commenté :

MyListAdaptater.java

```
package com.developpez.vue;

import java.util.List;
import java.util.Map;
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.CheckBox;
import android.widget.SimpleAdapter;

public class MyListAdapter extends SimpleAdapter
{
  private LayoutInflater  mInflater;

  public MyListAdapter (Context context, List<?
extends Map<String, ?>> data, int resource,
String[] from, int[] to)
  {
    super (context, data, resource, from, to);
    mInflater = LayoutInflater.from (context);

  }

  @Override
  public Object getItem (int position)
  {
    return super.getItem (position);
  }

  @Override
  public View getView (int position, View
convertView, ViewGroup parent)
  {
    //Ce test permet de ne pas reconstruire la
vue si elle est déjà créée
    if (convertView == null)
    {
      // On récupère les éléments de notre vue
      convertView = mInflater.inflate
(R.layout.list_detail, null);
      // On récupère notre checkBox
      CheckBox cb = (CheckBox)
convertView.findViewById (R.id.check);
      // On lui affecte un tag comportant la
position de l'item afin de
      // pouvoir le récupérer au clic de la
checkbox
      cb.setTag (position);
    }
    return super.getView (position, convertView,
parent);
  }
}
```

## 7. Conclusion

Et voilà, votre application est terminée ! Vous pouvez désormais créer des listes et contrôler les événements liés aux éléments situés à l'intérieur de celle-ci.

*Retrouvez l'article de Silvera David en ligne : Lien 18*

## Introduction à Xtext : créer son propre DSL

Le but de cet article est d'apprendre comment créer un DSL à travers le framework Java Xtext via Eclipse.

Nous allons commencer par donner quelques définitions, puis nous allons évoquer brièvement d'autres technologies connexes à Xtext ainsi que l'installation de cette dernière. Par la suite, nous allons apprendre à créer un projet Xtext et enfin, par un exemple simple, nous allons apprendre comment implanter la grammaire d'un DSL. Il serait nécessaire d'avoir quelques prérequis en théorie de langages formels pour aborder le chapitre 3 de cet article.

### 1. Introduction

#### 1.1. DSL ?

DSL est sigle de Domain-Specific Language (en anglais), c'est-à-dire Langage Spécifique au Domaine. Il s'agit généralement d'un « petit » langage de programmation dédié à un domaine d'activité spécifique. Il est conçu pour apporter des solutions aux problèmes soulevés dans un domaine particulier et donc est ainsi difficilement réutilisable dans d'autres domaines. C'est pourquoi ce type de langage de programmation se distingue des GPL (General Purpose Language) tels que Java, C, C++... dont le but de ces derniers est de proposer des solutions à tout type de problème solvable par ordinateur. Ce qui peut ne pas être la manière la plus optimale. Les concepts et les notations d'un DSL doivent alors être choisis pour être suffisamment proches des objets manipulés dans le domaine traité. Notons que les problèmes du domaine traité sont naturellement ceux pouvant être résolus par ordinateur.

#### 1.2. Xtext ? À quoi ça peut servir ?

Xtext est un framework développé sur Eclipse et permettant d'implémenter votre propre DSL textuel ou même de mettre sur pied un langage de programmation générique (un GPL). Il fonctionne sur une JVM (Machine Virtuelle Java) et est constitué de plusieurs API qui vous permettent de décrire les différents aspects du langage que vous voulez créer et propose une implémentation complète de ce langage. Notons que le langage développé sera alors un langage qui s'appuie et qui surcouche le langage Java. C'est-à-dire que les objets offerts par ce framework sont complètement des objets Java et le développement du compilateur de votre langage sera alors écrit en Java.

Xtext peut servir entre autres à proposer des solutions dans les domaines tels que : les systèmes embarqués, l'automobile, les appareils mobiles, l'automatique, les jeux vidéo, etc.

Notons aussi que les langages développés via Xtext peuvent selon son tutoriel être définis non seulement avec le langage Java, mais aussi d'autres langages tels que C, C++, C# objective C, Python et Ruby. De plus, une fois que le DSL est mis en œuvre , son utilisation devient indépendante de la JVM.

#### 1.3. Qu'offre Xtext par rapport aux technos Java les plus utilisées ?

D'après la littérature, il existe deux technologies Java très populaires pour la mise en œuvre de DSL ou de GPL. Ce sont : JavaCC et ANTLR.
JavaCC (Java Compiler Compiler) et ANTLR (ANother Tool for Language Recognition) sont ainsi des technos permettant de mettre en œuvre un langage de programmation et ont existé avant Xtext. Elles permettent à travers la grammaire que l'on a définie pour le langage, de générer automatiquement les objets (lexer, parser : constructeur de l'arbre syntaxique abstraite, contrôleur de type...) nécessaires pour sa mise en œuvre. Le développeur devait alors utiliser ces objets pour développer son compilateur ou l'interpréteur. Puis la phase de création de l'IDE du langage correspondait à une étape où le développeur devait se taper une grande partie du code à la main. Xtext est un framework innovateur intégré à Eclipse et qui offre une simplicité dans le développement de DSL et/ou GPL. Il surcouche ANTLR qui s' occupe de la fabrication des objets ci-dessus cités (lexer, parser...), mais aussi offre un éditeur Eclipse (dont nous devons continuer le développement) du nouveau langage avec déjà de nombreux opérateurs par défaut tels : la complétion de code, la coloration syntaxique, l'analyse syntaxique, etc.

Pour résumer, nous disons que Xtext est un framework libre développé sur Eclipse et qui intègre tout ce qui est nécessaire pour le développement d'un DSL ou d'un GPL de façon complète : c'est-à-dire de la spécification de la grammaire du DSL, en passant par le développement de son compilateur, jusqu'au développement de l'IDE Eclipse du DSL. De plus, puisque Xtext utilise EMF (Eclipse Modeling Framework), il offre des packages très riches que nous n'avons qu'à exploiter dans nos développements. Xtext offre également une grammaire initiale que l'on peut étendre (utiliser) dans la définition de la nôtre. Cette grammaire initiale offre des éléments importants comme la gestion des espaces, des retours chariot, tabulation, des chaines de caractères, des entiers, etc.

N.B. : il existe plusieurs autres technologies concourant au développement de langages de programmation telles que : lex/flex, yacc/bison (en C) et aussi JFlex/JCup, SableCC

ou encore Tatoo, etc.

### 1.4. Installation d'Eclipse Xtext

L'installation d'Eclipse Xtext est simple. Il suffit de télécharger le fichier zip sur le site itemis des développeurs d'Xtext à l'adresse http suivante : Lien 19. Notons qu'il existe plusieurs versions correspondant à différents systèmes d'exploitation. Téléchargez donc la version qui correspond à votre cas. Lorsque le téléchargement est terminé, il suffit de copier le fichier zip sur votre disque dur à l'endroit où vous désirez. Après l'avoir dézippé, dans le répertoire qui se crée, on constate la présence de plusieurs dossiers et fichiers parmi lesquels eclipse.exe. Il suffit alors de double-cliquer sur cette icône pour démarrer Eclipse Xtext.

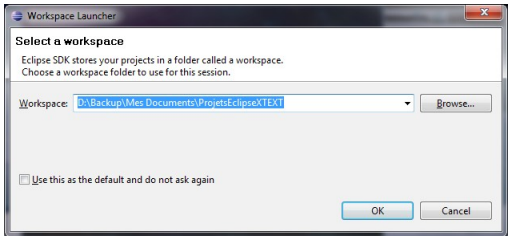
Notons pour finir que l'on peut créer un raccourci de cet exécutable (sur son bureau par exemple) en cliquant à droite sur *eclipse.exe*, puis en choisissant *envoyer vers* et enfin *bureau (créer un raccourci)*. Ceci pour s'affranchir d'une perte de temps lors du démarrage d'Eclipse Xtext.

## 2. Premier projet avec Xtext

Nous allons apprendre à créer un projet Xtext sous Eclipse Helios. Nous utilisons la version 1.0.2 de Xtext.

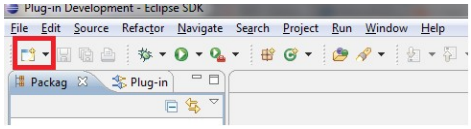
2.1- Double-cliquer sur l'icône *eclipse.exe* dans le répertoire correspondant ou sur le raccourci que vous avez créé sur votre bureau.

2.2- la fenêtre suivante apparaît :



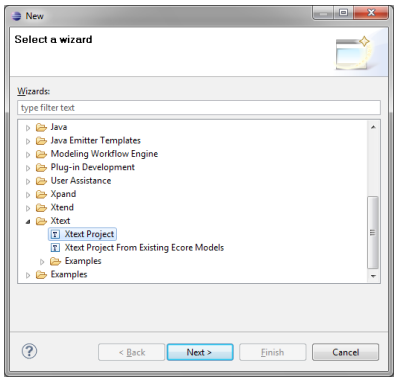

Il faut indiquer à Eclipse Xtext, le répertoire de vos projets (le workspace). Un exemple ici est le répertoire D:\Backup\Mes Documents\ProjetsEclipseXTEXT. Après avoir choisi votre workspace, cliquer sur OK.

2.3- Dans la fenêtre qui s'ouvre, cliquer sur l'icône à l'extrême gauche pour ouvrir la fenêtre de choix du type de projet à créer. Dans l'aperçu suivant, cette icône est encadrée d'un rectangle rouge :



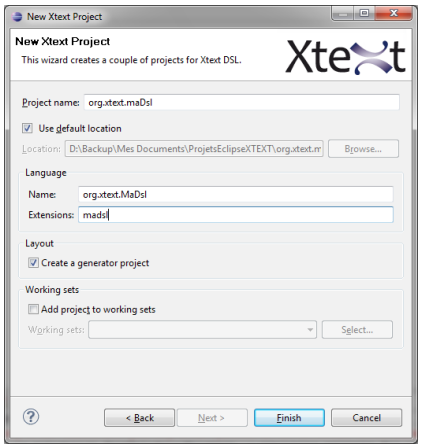

Notons que l'on peut aussi cliquer sur la petite flèche à côté de cette icône pour dérouler un menu dans lequel on va choisir l'onglet *Project*. On peut aussi réaliser l'opération précédente en passant par la procédure suivante : *File-->New-->Project*.

2.4- Dans la fenêtre du choix du type de projet qui s'ouvre, dérouler l'arbre nommé *XText*, puis choisir *Xtext project* et enfin cliquer sur next :

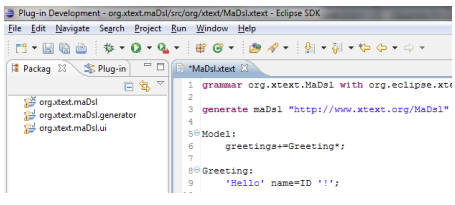



2.5- Dans la fenêtre qui apparaît, donner un nom à votre projet dans le champ *project name*. Exemple : *maDsl*. ( N.B. : ce nom doit commencer par une lettre minuscule). Puis, donner un nom à votre langage. Exemple : *MaDsl* (N.B. : ce nom doit commencer par une lettre majuscule). Enfin, nommer l'extension des fichiers où seront enregistrés les programmes de votre langage. Exemple : *madsl*. Cliquer sur *finish*, pour terminer la création. Il est mieux de créer notre projet dans le package de base *org.xtext* fourni par Xtext. Ce qui fait que lors de la saisie du project name, nous devons effacer ce qui est écrit par défaut et saisir comme nom de projet : *org.xtext.maDSL*. Cela est pareil pour le nom de votre langage. De plus, il vaut mieux laisser les cases à cocher telles qu'elles sont. Enfin, il serait quand même intéressant de choisir des noms significatifs pour des projets réels et sérieux et non des choses comme *maDsl*.

Voici l'aperçu de cette fenêtre :




2.6- Dans la fenêtre Eclipse qui apparaît, on voit le projet Xtext qui est créé et qui est composé de trois parties (org.xtext.maDsl, org.xtext.maDsl.generator et enfin org.xtext.maDsl.ui).




Notre premier projet Xtext a été créé avec succès. Nous allons voir dans la section suivante, comment implanter la grammaire d'un DSL en prenant un exemple simple.

## 3. Implanter une grammaire pour votre DSL

Nous allons dans cette section apprendre à développer la grammaire d'un DSL sous Eclipse Xtext. Pour cela, il nous faut partir d'un exemple. Nous allons donc dans le paragraphe suivant proposer un exemple simple pour comprendre comment on procède. De plus, notons que les grammaires écrites sous Xtext s'appuient sur la forme étendue de Backus-Naur (EBNF).

### 3.1. Description de notre DSL

Supposons que nous voulions construire un DSL qui concerne le domaine de la statistique (les statisticiens). Supposons que nous avons des statisticiens qui étudient des populations de tout genre et qui au terme de leurs travaux disposent :

- des différentes modalités de la population étudiée ;
- des effectifs de chaque modalité.

Ils aimeraient faire des études sur cette population qui peuvent d'ailleurs être très complexes. Pour faire simple, supposons que leurs études se limitent au calcul de la **moyenne**, de la **variance**, de **l'écart type** et du **mode** de cette population.

Pour rester dans un contexte simpliste, nous allons supposer que les modalités de la population sont quantitatives et ne peuvent être que des entiers et/ou des réels (et donc pas de modalité qualitative ou encore par intervalle). De plus, nous supposons que le nombre de modalités ne doit pas être très grand au point de ne pas pouvoir être saisissables à la main.

Exemple : supposons qu'une étude statistique vise à comprendre des phénomènes sur les notes d'un examen d'informatique d'une salle de classe d'un effectif total de 20 étudiants.

Voici dans le tableau suivant ce qui pourrait être recensé comme données des statisticiens :

| Modalités (notes) | 08 | 11 | 12 | 14 | 15 | Total |
|---|---|---|---|---|---|---|
| Effectifs (étudiants) | 5 | 6 | 4 | 3 | 2 | 20 |

Notons que si nous avons une série statistique de modalités $x_i$ et d'effectifs $n_i$, tels que l'effectif total est $n = \sum_{i=1}^{n} n_i$, alors :

La moyenne est égale à : $$\bar{X} = \frac{1}{n} \cdot \sum_{i=1}^{n} n_i x_i$$

La variance notée V est égale à : $$\frac{1}{n} \cdot \sum_{i=1}^{n} (x_i - \bar{X})^2$$

L'écart type est égal à la racine carrée de la variance V.

Le mode est égal à la modalité ayant le plus grand effectif.

Nous voulons écrire des programmes statistiques simples qui vont permettre de calculer ces valeurs à chaque fois pour une série statistique donnée.

Nous voulons que les programmes de notre DSL aient la structure simple suivante :

```
program nom_du_programme
 begin
   mod : // liste des modalités séparées par des
virgules et se terminant par un point-virgule
   eff : // liste des effectifs respectifs
séparés par des virgules et se terminant par un
point-virgule
   return // suivi de l'opération que l'on veut
effectuer et termine par un point-virgule.
 end
```

**Illustration :**

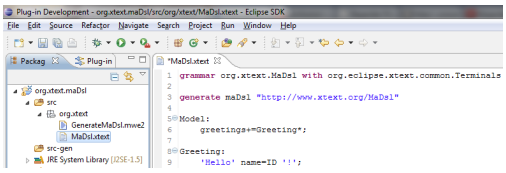
```
program calculMoyenne
 begin
   mod : 08, 11, 12, 14, 15 ;
   eff : 5, 6, 4, 3, 2 ;
   return moyenne ;
 end
```

Notons que contrairement à cet exemple hyperacadémique, dans la réalité, les langages peuvent être divers et complexes. Notre but étant juste d'utiliser cet exemple pour montrer comment travailler avec Xtext.

### 3.2. Développement de la grammaire du DSL sous Xtext

Pour écrire la grammaire de notre DSL, nous devons ouvrir le premier projet parmi les trois qui ont été précédemment créés (dans notre cas, il s'agit du projet *org.xtext.maDsl*), puis double-cliquer sur le fichier ayant l'extension *.xtext* (dans notre cas *MaDsl.xtext*). En général, après création de votre projet Xtext, ce fichier s'ouvre automatiquement. Au cas où il ne s'ouvre pas ou s'ouvre mal, il faut l'ouvrir manuellement.

Aperçu :

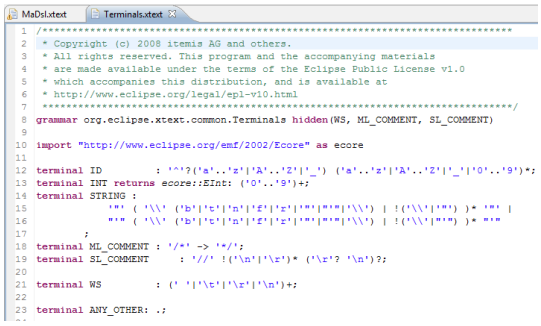


Avant de commencer à écrire notre grammaire, on constate que le fichier *MaDsl.xtext* s'ouvre avec une grammaire exemple. Nous allons profiter de cela pour gagner quelques prérequis avant de nous lancer.

- La première ligne : grammar org.xtext.MaDsl with org.eclipse.xtext.common.Terminals représente la déclaration de notre grammaire nommée *MaDsl* et le fait qu'elle étende (qu'elle utilise) la grammaire initiale généreusement offerte par Xtext (*org.eclipse.xtext.common.Terminals*). Cette grammaire offerte par Xtext déclare pour nous, un ensemble d'éléments importants et d'ailleurs

souvent très nuisant lorsque nous devons le faire par nous-mêmes. Nous n'aurons alors dans notre propre grammaire qu'à utiliser les éléments qu'elle nous offre sans nous poser de question. Les éléments offerts par cette grammaire initiale de Xtext sont de façon non exhaustive : les chaines de caractères (représentés par le terminal ID), les entiers (représentés par le terminal INT), les espaces, tabulations, retour chariot, les commentaires, etc. Notons que l'on n'est pas obligé d'utiliser cette grammaire initiale de Xtext, mais nous pensons qu'elle est importante pour écrire des grammaires simples à comprendre et rapidement.

Pour voir à quoi ressemble cette grammaire offerte par Xtext, maintenir la touche *Ctrl* enfoncée, et cliquer sur le texte suivant : *org.eclipse.xtext.common.Terminals* dans votre fenêtre Eclipse.
Aperçu :




- La deuxième ligne : generate maDsl "http://www.xtext.org/MaDsl" est très importante pour Xtext pour la génération des objets importants pour le développement du langage et qui sont issus de la grammaire. Mais nous n'entrons pas dans les détails ici. Toujours laisser cette deuxième ligne intacte.
- Puis nous avons les lignes suivantes qui représentent une ébauche de la façon dont nous devons écrire nos grammaires :

```
Model                    :
  greetings+=Greeting*          ;
Greeting:
  'Hello' name=ID '!';
```

Cette grammaire permet d'écrire zéro ou une liste de phrases du genre : Hello world !, Hello Georges !, etc. dans un IDE dédié au langage régi par cette grammaire. Dans cette grammaire, Model et Greeting représentent des non terminaux. Model représente l'axiome de la grammaire parce qu'il s'agit du tout premier non terminal, la racine. On voit dans cette grammaire que Model se dérive en *greetings+=Greeting** et Greeting à son tour se dérive en *'Hello' name=ID '!'* . Nous allons voir ci-dessous ce que représentent greetings et name. Dans la règle de production *Model: greetings+=Greeting**, l'étoile qui est à droite de Greeting signifie que Model se dérive zéro ou plusieurs fois au non terminal Greeting. Pour cela on utilise le symbole += pour marquer cette dérivation multiple. Si par exemple, Model se dérivait en une seule fois en Greeting, on écrirait : *Model : greetings=Greeting*. Puis, nous avons Greeting qui se dérive en une phrase qui commence par *Hello* suivi d'une chaine de caractères et se termine par un point d'exclamation. Notons qu'une règle de production sous Xtext se termine toujours par un point-virgule.

Lorsque Xtext voit ainsi une telle grammaire écrite, il génère entre autres objets (lexer, parser...), des classes Java (interfaces et implémentations) correspondant à chaque non terminal ayant fait intervenir une variable (cf. greetings, name). Ce sont ces classes encapsulées comme objets dans l'arbre syntaxique généré par Xtext, que nous utiliserons pour écrire le compilateur ou l'interpréteur du langage. Nous allons voir à la fin de ce document comment générer ces classes. À l'intérieur de chacune de ces classes, nous disposons des getters et des setters correspondant à la structure des règles de production de la grammaire. Pour l'exemple précédent, Xtext va nous générer les interfaces des classes Model et Greeting qui ressemblent à ceci :

| | |
|---|---|
| `public Model extends Eobject{`<br><br>`EList<Greeting> getGreetings();`<br><br>`}` | `Public Greeting extends EObject{`<br><br>`String getName();`<br>`void setName(String name);`<br><br>`}` |

On voit bien que la classe Model permet de récupérer une liste d'objets Greeting, alors que la classe Greeting permet de récupérer la chaine de caractères name qui avait alors une valeur inconnue dans la grammaire. De même, c'est la variable greetings déclarée dans la grammaire qui est utilisée pour fabriquer le getter de la classe Model (getGreetings()). EList représente le type List de java.util sous Xtext (cf. EMF) et est manipulé de la même façon en utilisant un itérateur. EObject représente le model de nos programmes, l'arbre syntaxique abstrait...

Lorsque nous voulons donc écrire une grammaire sous Xtext, si nous voulons qu'un non terminal soit représenté par une classe Java lors de la génération, nous devons dans la définition de sa règle de production utiliser des variables. Xtext va donc utiliser ces variables pour générer des getters et des setters.

Exemple : soit une règle de production (r).

1. Si (r) s'écrit **X : y=ID ;** alors, Xtext générera une classe X contenant un getter et un setter pour la variable y de type String (cf. terminal ID). De même si on remplace ID par un non terminal quelconque (ex : Z), alors y sera de type Z.
2. Si (r) s'écrit **X : y?='bonjour' ;** alors, la variable y est de type booléen et Xtext va générer une classe X avec une méthode de test booléenne sur y qui vaut true lorsque le mot *bonjour* est saisi et false sinon, puis une méthode setter sur y.
3. Si (r) s'écrit **X : (y=Z)?** alors cette règle signifie que X se dérive zéro ou une fois en Z. C'est-à-dire que X se dérive en Z ou en l'élément vide (ou en rien). Xtext génère une classe X exactement comme au point 1.
4. Si (r) s'écrit **X : (y+=Z)*** équivalent à X : y+=Z* alors cette règle signifie que X se dérive zéro ou

plusieurs fois en Z. La variable y est de type une liste de Z. Xtext va générer une classe X contenant le getter de y qui représente une liste de Z. La parenthèse permet de factoriser l'écriture, parce que (r) peut par exemple s'écrire, X : (y+=Z | p+=W)* où | représente l'opérateur de choix. C'est-à-dire que X se dérive zéro ou plusieurs fois en Z ou en W et pas en Z et W en même temps.

5. Si (r) s'écrit **X : (y+=Z)+** alors, c'est exactement la même chose qu'au point 4. à la différence que X se dérive ici une ou plusieurs fois en Z et non plus zéro ou plusieurs fois.

Notons que nous pouvons aussi combiner certains de ces opérateurs. Par exemple, si nous écrivons la règle X : (y=Z) (p=W)? (g+=L)*, cela voudrait dire que X est un non terminal qui se dérive en une seule fois en Z suivi d'un W optionnel et enfin de zéro ou une liste de L.

Maintenant que nous avons les prérequis pour écrire notre grammaire, nous devons saisir dans le fichier *MaDsl.xtext* la grammaire suivante :

```
grammar org.xtext.MaDsl with
org.eclipse.xtext.common.Terminals

generate maDsl "http://www.xtext.org/MaDsl"

import "http://www.eclipse.org/emf/2002/Ecore" as
ecore

PROGRAMME :
   (pream=PREAMBULE deb=DEBUT mod=MODALITES
eff=EFFECTIFS ret=RETURN end=END;)? ;

PREAMBULE :
   'program' nomProgramme=ID ;

DEBUT :
   'begin' ;

MODALITES :
   'mod' ':' premiereModalite=REEL (',' 
autreModalite+=REEL)+)? ';' ;

EFFECTIFS :
   'eff' ':' premierEffectif=REEL (',' 
autreEffectif+=REEL)+)? ';' ;

RETURN :
   'return' resultat=RETOUR ';' ;

RETOUR :
   'ecarttype'|'moyenne'|'variance'|'mode';

END :
   'end' ;

REEL :
   reel=DOUBLE ;

terminal DOUBLE returns ecore::EDouble: '-'?INT
('.' INT)? ;
```

N.B. : Faire attention au copier/coller dans ce fichier *.xtext* qui peut souvent générer des erreurs. Nous allons maintenant, ligne par ligne, essayer de décrypter cette grammaire pour la comprendre. Les deux premières lignes de cette grammaire ont déjà été expliquées.

1.

```
import "http://www.eclipse.org/emf/2002/Ecore" as
ecore
```

http://www.eclipse.org/emf/2002/Ecore est un package EMF contenant aussi le package EPackage de gestion des types de données (cf. EList). Cette ligne signifie donc qu'on importe ce package dans notre grammaire sous le nom *ecore*. L'utilisation de ce package dans notre grammaire est faite à la dernière ligne du fichier *MaDsl.xtext*.

2.

```
PROGRAMME :
   (pream=PREAMBULE deb=DEBUT mod=MODALITES
eff=EFFECTIFS ret=RETURN end=END;)? ;
```

Il s'agit de la première règle de production de notre grammaire. PROGRAMME est donc l'axiome de notre grammaire, la racine de l'arbre syntaxique. Cette règle stipule que notre programme se dérive en cinq parties : un PREAMBULE, une déclaration de DEBUT, une déclaration des MODALITES, une déclaration des EFFECTIFS, un RETURN pour retourner le résultat du calcul souhaité et une déclaration de fin de programme END. Tous ces non terminaux seront à leur tour définis dans les lignes qui suivent. Notons que pour bien assurer la lisibilité de notre grammaire, nous avons décidé d'écrire en majuscules les non terminaux et les variables qu'utilise Xtext dans la génération des classes en minuscules. Exemple : l'écriture **pream=PREAMBULE** signifie que la classe PROGRAMME qui sera générée, contiendra un getter et un setter de la variable *pream* de type *PREAMBULE*. L'opérateur d'optionalité « ? » est utilisé ici pour permettre à ce qu'un PROGRAMME se dérive en ces cinq parties ou en rien. Ceci afin de permettre qu'un programme vide soit aussi un programme valide. Si nous l'omettons, nous allons constater que lorsque nous ouvrons « l'IDE » de notre langage (cf partie 3.4), il s'ouvre avec une erreur. Ceci parce qu'il s'attend spontanément à voir ces cinq parties saisies. Vous pouvez faire l'expérience en l'omettant.

3.

```
PREAMBULE :
   'program' nomProgramme=ID ;
```

Cette ligne stipule que notre non terminal PREAMBULE précédent se dérive en un terminal appelé program (qui représente ainsi un mot-clé du langage. Et par généralisation, tous les mots entre quottes sont des mots-clés de notre langage) suivis d'une chaine de caractères (de valeur à saisir par l'utilisateur) qui va représenter le nom de notre programme.

4.

```
DEBUT :
   'begin' ;
```

Ici le non terminal DEBUT se dérive en un mot-clé appelé *begin*. Cela veut dire que dans nos programmes, après avoir déclaré le préambule, nous devons obligatoirement écrire le mot *begin*.

5.

```
MODALITES :
   'mod' ':' premiereModalite=REEL (',' autreModalite+=REEL)+)* ';';
```

Cette ligne signifie que le non terminal MODALITES se dérive :

- en un terminal appelé *mod*, suivi du terminal *:* (un deux-points ). *mod* et le deux-points n'a pas été mis dans la même paire de quottes pour rendre le langage flexible. Car faire ainsi, permet à l'utilisateur de mettre autant d'espaces entre *mod* et le deux-points ;
- suivi d'une liste de réels représentant les différentes modalités de la série statistique. Sachant qu'une étude peut donner lieu à une seule modalité, nous sommes amenés à déclarer une première modalité obligatoire à écrire par l'utilisateur (premiereModalite=REEL), suivi d'une liste optionnelle d'autres modalités séparées par des virgules. Ceci par l'écriture : ((',' autreModalite+=REEL)+)*. REEL est un non terminal représentant les nombres réels et doit être défini plus tard ;
- tout cela enfin se terminant par un point-virgule.

6.

```
EFFECTIFS :
   'eff' ':' premierEffectif=REEL (',' autreEffectif+=REEL)+)* ';' ;
```

Cette ligne est pareille en explication que la précédente. Mais concerne plutôt les effectifs correspondant à chaque modalité.

7.

```
RETURN :
   'return' resultat=RETOUR ';' ;
```

Cette règle stipule que le non terminal RETURN se dérive en un mot-clé *return* suivi d'un non terminal RETOUR et enfin d'un point-virgule. Nous allons voir dans ce qui suit en quoi se dérive RETOUR.

8.

```
RETOUR :
   'ecarttype'|'moyenne'|'variance'|'mode';
```

Le non terminal RETOUR se dérive en quatre mots-clés à prendre chacun au choix. Soit *ecarttype*, soit *moyenne*, soit... Cela veut dire que dans nos programmes après avoir renseigné les modalités et les effectifs, si nous écrivons *return moyenne*, le programme est sensé calculer la moyenne arithmétique de cette série statistique et nous retourner le résultat.

9.

```
END :
   'end' ;
```

Puis, nous avons la règle qui fait appel au non terminal END qui se dérive en la chaine de caractères (mot-clé) *end* pour clôturer le programme.

Il nous reste maintenant à définir les règles utilisées pour définir les nombres réels dans notre grammaire.

10.

```
REEL :
   reel=DOUBLE ;
```

Cette règle stipule que le non terminal REEL se dérive en un terminal appelé DOUBLE qui représente en fait l'ensemble des nombres réels que nous allons définir. Nous allons voir dans la dernière ligne suivante comment cela est défini sous Xtext.

11.

```
terminal DOUBLE returns ecore::EDouble: '-'?INT ('.' INT)? ;
```

Pour définir un ensemble de données à partir d'un autre sous Xtext, la procédure à suivre est comme dans le tableau précédent. Cette ligne est rattachée à la ligne « import "http://www.eclipse.org/emf/2002/Ecore" as ecore », où *ecore* représente le nom de ce package dans notre grammaire. Comme nous voulons que cet ensemble de données s'appelle DOUBLE, nous l'écrivons entre les mots-clés : terminal et returns. Puis nous écrivons le terme ecore::EDouble qui permet de convertir l'écriture '-'?INT ('.' INT)? en réel. EDouble est une instance de EDataType de EMF qui contient quasiment, sinon tous les types de données que nous connaissons. L'écriture '-'?INT ('.' INT)? représente ainsi les réels dans notre langage. Cette écriture signifie qu'un nombre réel peut ou non commencer par un signe négatif, suivi d'un entier, et enfin suivi d'une partie optionnelle qui représente la partie décimale de ce réel. Cette partie est optionnelle parce qu'un réel est avant tout un entier.

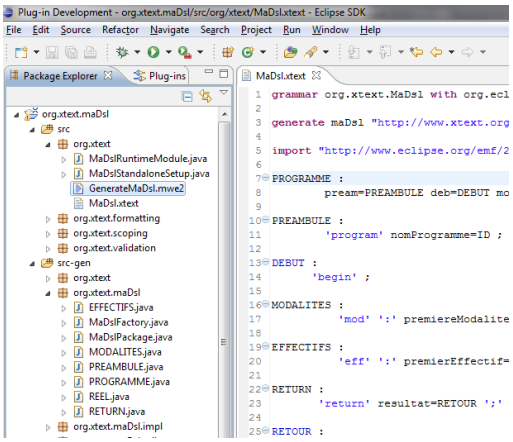
Nous avons ainsi terminé l'écriture de la grammaire de notre DSL destiné au domaine de la statistique. Nous allons maintenant apprendre pour terminer cet article à générer les classes Java correspondant aux éléments de notre grammaire, puis comment exécuter notre projet pour éditer un programme.

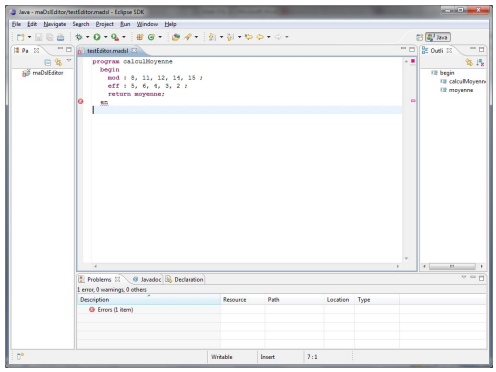
### 3.3. Génération de code

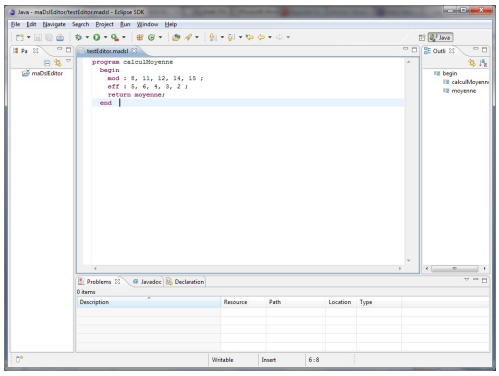
Pour générer les classes Java liées aux non terminaux de notre grammaire ainsi que le parseur et autres objets Java de notre DSL, voici la procédure à suivre : dans le projet *org.xtext.MaDsl*, cliquer à droite sur le fichier *GenerateMaDsl.mwe2*, puis dans le menu qui s'ouvre, choisir *Run As* et enfin cliquer sur *1 MWE2 Workflow*. Si la compilation se passe bien, c'est que votre grammaire ne comporte aucune erreur.

Voici un aperçu des classes générées à l'issue de l'opération précédente :

Les classes MaDslRuntimeModule.java, MaDslStandaloneSetup.java ont été automatiquement générées ainsi que celles du dossier *src-gen/org/xtext/maDsl*. On peut reconnaître qu'elles ont le même nom que les non terminaux utilisés dans la grammaire. Il s'agit des interfaces dont nous parlions précédemment. Leur implémentation se trouve dans le package *org.xtext.maDsl.impl*.





## 3.4. Exécution du projet et test de saisie d'un programme

Pour exécuter le projet, nous devons cliquer à droite sur le premier des trois projets (*org.xtext.maDsl*), dans le menu qui s'ouvre, choisir *Run As*, puis *1 Eclipse Application*. Une nouvelle fenêtre Eclipse va se lancer et s'ouvrir. Dans cette nouvelle fenêtre Eclipse, nous allons créer un nouveau projet de type général (*File-->Project-->New-->General-->Project*). Renseigner le nom du projet dans la fenêtre qui s'ouvre (ex : *maDslEitor*). Une fois ce projet créé, cliquer droit sur ce projet, puis choisir *New* dans le menu qui apparait, choisir *File* dans le nouveau menu qui apparait également. Une fenêtre s'ouvre, dans cette fenêtre saisir dans le champ *File name* le nom d'un fichier du projet *maDslEditor* avec l'extension *.madsl* (exemple : *testEditor.madsl*). Cliquer sur OK. Eclipse va se rendre compte qu'il s'agit d'un fichier d'édition des programmes votre DSL et va nous demander s'il importe Xtext dans ce projet. Choisissons *Yes*. Une fois le fichier apparu dans notre projet *maDslEditor*, double-cliquer pour l'ouvrir. Il représente pour l'instant l'IDE de notre DSL. Nous pouvons maintenant saisir des programmes de notre DSL. Nous allons constater qu'Xtext souligne les erreurs syntaxiques commises dans ces programmes. Ce qui veut dire qu'il embarque dans cet éditeur un analyseur syntaxique pour notre langage.

Voici pour finir l'aperçu de l'édition de deux programmes, l'un comportant une erreur syntaxique et l'autre sans erreur. On peut aussi remarquer la coloration syntaxique offerte.

## 4. Conclusion

Au terme de ce travail, nous avons abordé la notion de DSL et nous avons appris comment mettre sur pied un DSL à travers le framework Xtext. Nous avons également évoqué d'autres technologies aidant à la création des DSL. Par un exemple banal, nous avons appris à écrire une grammaire d'un langage sous Xtext et comment exécuter notre projet jusqu'à la réalisation de tests pour voir si ce projet Xtext réalise ce que nous attendons du langage.

Ce qui m'a motivé dans l'écriture de cet article vient du fait de la très faible documentation sur l'utilisation de ce framework balaise, si ce n'est le User Guide de Xtext lui-même qui est d'ailleurs très long et en anglais. J'ai alors décidé de rédiger cet article afin de partager ma petite expérience avec ceux qui ont des projets pouvant porter dans le domaine de la réalisation d'un DSL ou d'un GPL.

## 5. Perspectives

Dans de prochains articles, nous verrons comment faire pour utiliser les classes et autres objets que nous génère Xtext pour écrire le compilateur ou l'interpréteur de notre langage. Puis, nous verrons si nous parvenons à le réaliser nous-mêmes, comment développer la partie ui de notre projet Xtext (ex : org.xtext.maDsl.ui) afin de créer le propre IDE de notre langage.

## 6. Références

Xtext User Guide : Lien 20.

*Retrouvez l'article de Georges Kemayo en ligne : Lien 21*

## Doctrine2 et les fixtures dans Symfony2

Lors de tests, il est souvent pratique de repartir d'une base de données vierge ; or, une base de données vierge est… vierge de données. Les fixtures permettent, lors de la création de la base, d'ajouter des données, les fixtures.
Elles ne sont pas supportées de base dans Doctrine 2, au contraire de la première version, il faudra donc installer un bundle supplémentaire.

### 1. Installation de Doctrine Data Fixtures

Dans l'édition Standard de Symfony2, l'appel du script bin/vendors.php lance la mise à jour et le téléchargement d'une série de dépôts Git, précisés dans un fichier de dépendances. Par défaut, ce fichier est rempli avec le framework Symfony2 et les paquets nécessaires pour former l'édition standard. On peut cependant y ajouter de nouveaux dépôts à télécharger. C'est la méthode qui sera utilisée ici.

Tout d'abord, dans le fichier de dépendances deps, il faut ajouter ces quelques lignes :

```
[doctrine-fixtures]
    git=http://github.com/doctrine/data-fixtures.git

[DoctrineFixturesBundle]
    git=http://github.com/symfony/DoctrineFixturesBundle.git
    target=/bundles/Symfony/Bundle/DoctrineFixturesBundle
```

Elles indiquent qu'il y a deux nouvelles dépendances à télécharger. Il faut ensuite lancer la mise à jour des vendors afin de les installer effectivement et, le cas échéant, de mettre à jour votre installation de Symfony2 :

```
bin/vendors install
```

Le bundle est maintenant installé, il faut que l'application l'inclue dans son noyau (kernel) et autoloading.

Il faut ajouter le nouvel espace de noms dans ceux connus de Symfony2 ; cependant, on ne peut pas ajouter cette entrée n'importe où : si on met le nouvel espace (Doctrine\Common\DataFixtures) après l'actuel de Doctrine (Doctrine\Common), toute requête sur cet espace finira en erreur, Symfony2 ira d'abord chercher dans l'espace commun de Doctrine.

Dans le fichier app/autoload.php :

```
$loader->registerNamespaces(array(
    // ...
    'Doctrine\\Common\\DataFixtures' => __DIR__.'/../vendor/doctrine-fixtures/lib',
    'Doctrine\\Common' => __DIR__.'/../vendor/doctrine-common/lib',
    // ...
));
```

Ensuite, il faut charger ce bundle. Pour ce faire, dans le fichier app/AppKernel.php :

```
public function registerBundles()
{
    $bundles = array(
        // ...
        new Symfony\Bundle\DoctrineFixturesBundle\DoctrineFixturesBundle(),
        // ...
    );
    // ...
}
```

### 2. Première fixture

Les fixtures sont simplement des classes dont la méthode load() est exécutée. Ces classes doivent se trouver dans le dossier DataFixtures/ORM de votre bundle si vous utilisez l'ORM, DataFixtures/ODM pour l'ODM (MongoDB, CouchDB ou autres).

Pour le nommage, il est préférable d'utiliser la convention suivante : LoadEntityData, où Entity est le nom de l'entité.

On met dans le fichier un contenu similaire à ce qui suit :

```
<?php

namespace ...\...\DataFixtures\ORM;

use Doctrine\Common\DataFixtures\FixtureInterface;
use ...\...\Entity\En;

class LoadEnData implements FixtureInterface
{
    public function load($manager)
    {
        $en = new En();
        // Quelques modifications sur l'entité pour qu'elle serve à quelque chose

        $manager->persist($en);
        $manager->flush();
    }
}
```

Une fois ce fichier créé, rempli et adapté, il faut importer ces fixtures en lançant la commande suivante :

```
app/console doctrine:fixtures:load
```

En fonction de la base de données sous-jacente, il faudra possiblement adapter la commande :

```
app/console doctrine:mongodb:fixtures:load
```

Remarquons la flexibilité que ce système offre par rapport à son équivalent pour Doctrine dans sa première version : on peut générer très facilement des fixtures dynamiquement, en fonction des résultats d'une requête Google, par exemple.

### 3. Ordre de chargement

Il n'est pas possible en l'état de se baser sur l'existence ou non d'entités en base, alors que la définition de nouvelles entités requiert l'existence d'autres (on ne peut pas créer un article sans catégorie associée sur un blog, notamment).

On utilise alors un ordre de chargement : les fixtures seront chargées selon l'ordre de chargement indiqué. On le spécifie en définissant une méthode publique getOrder() dans le fichier de fixtures. Il renverra un entier : le fichier de fixtures ayant l'entier le plus bas sera chargé en premier, suivi des autres dans l'ordre croissant.

```
<?php

namespace ...\...\DataFixtures\ORM;

use
Doctrine\Common\DataFixtures\FixtureInterface;
use ...\...\Entity\En;

class LoadEnData implements FixtureInterface
{
    public function load($manager)
    {
        $en = new En();
        // Quelques modifications sur l'entité
pour qu'elle serve à quelque chose

        $manager->persist($en);
        $manager->flush();
    }

    public function getOrder()
    {
        return 1;
    }
}
```

### 4. Pour insérer plusieurs entités à la fois

Il n'est pas rare de devoir insérer plusieurs entités à la fois, parfois même de créer les entités en fonction d'un résultat externe. Dans ces cas, il est probablement plus pratique d'externaliser la création de l'entité et sa persistance comme ceci :

```
<?php

namespace ...\...\DataFixtures\ORM;

use
Doctrine\Common\DataFixtures\FixtureInterface;
use ...\...\Entity\En;

class LoadEnData implements FixtureInterface
{
    private $manager;

    public function load($manager)
    {
        $this->manager = $manager;

        $this->generateEntities();

        $this->manager->flush();
    }

    public function getOrder()
    {
        return 1;
    }

    private function generateEntities()
    {
        $vars = array('cats', 'dogs');

        foreach($vars as $v)
        {
            $this->newEntity($v);
        }
    }

    private function newEntity($param)
    {
        $en = new En();
        $en->setParam($param);
        $this->manager->persist($en);
    }
}
```

À noter que la fonction generateEntities() n'est pas requise, elle permet simplement d'avoir des fonctions courtes et qui n'ont qu'une seule utilité : load() initialise et lance la génération des entités et valide leur entrée en base de données ; generateEntities() génère les données nécessaires à la création de ces entités ; newEntity() crée l'entité.

*Retrouvez l'article de Thibaut Cuvelier en ligne : Lien 22*

## L'indexation des données dans le monde du Web sémantique

Le Web des données n'est pas très différent du Web des documents : pour que quelqu'un vienne voir quelles informations sont disponibles, il faut que cette personne puisse les trouver. Un grand annuaire n'est pas possible, car il sera vite dépassé par la quantité de données à indexer.
Un vocabulaire vient ici à la rescousse : VoID, le vocabulaire des ensembles de données interconnecté(e)s. Il permet d'indexer des sources de triplets RDF en décrivant les informations disponibles.

### 1. Un bref historique

Retour au seizième siècle. Début de l'imprimerie. Les livres commençaient à se répandre. Les érudits avaient de nouvelles sources.

Cependant, tout n'allait pas forcément pour le mieux dans le meilleur des mondes : outre le fait que tout le monde n'avait pas accès à la connaissance, il fallait encore savoir... ce qui était disponible.

1595. Leiden, Pays-Bas. *Nomenclator.*

Le premier catalogue imprimé.

Enfin, on pouvait savoir à Berlin ce que contenait une bibliothèque. Sans s'y trouver physiquement. Sans envoyer de demande par courrier postal.

D'autres bibliothèques ont suivi le mouvement, bien évidemment, pour ne pas se retrouver entièrement dépassées. D'autres catalogues imprimés parurent, furent distribués un peu partout dans le monde. On pouvait alors comparer plusieurs bibliothèques : vaut-il mieux aller à Sodome ou à Gomorrhe pour se documenter sur les découvertes de Christophe Colomb ?

Le Web des données se situe dans la même position : beaucoup de données sont disponibles, mais on ne sait pas qu'elles existent. Pour pallier ce problème, on a créé divers systèmes qui vont être décrits dans cet article. On va chercher à déclarer au monde entier qu'une source de données existe (généralement, un *SPARQL end point*, cela peut aussi bien être un fichier de triplets RDF sérialisés d'une manière ou d'une autre (À noter qu'un SPARQL endpoint *n'est pas* un fichier RDF. Les deux ont le même objectif : contenir des triplets RDF. Le premier a pour objectif d'effectuer des requêtes sur ces triplets, tandis que le second ne cherche qu'à les stocker sous une forme propre à l'échange de données.)), ce qu'elle contient (quels sont les sujets des données disponibles). Tout ce qu'il faut est un vocabulaire commun pour décrire ces sources de données.

### 2. L'expérience acquise du Web des documents

Le Web tel qu'actuellement connu, le Web des documents, permet lui aussi de faire des recherches, de trouver de l'information, grâce aux moteurs de recherche. Qu'est-ce qui a bien fonctionné dans ce système ? Quelles pistes intéressantes seraient à reprendre pour le Web des données ?

Il n'y a pas une seule et unique source de contenu. Chacun publie son contenu dans son coin, parfois sans se soucier qu'il soit facilement trouvable par d'autres. Il n'est pas obligatoire de présenter chaque nouvelle page, chaque modification à un organisme central. Par contre, il y a plusieurs moteurs de recherche, chacun tente de présenter son site Web de telle sorte qu'il soit bien indexé par ces moteurs, toute une industrie parallèle s'est développée sur ce sujet précis.

Cette solution a pu se développer à l'échelle actuelle du Web. Le créateur de contenu « annote » ses créations pour que les moteurs les retrouvent, les indexent, en prennent grand soin. Les moteurs, quant à eux, effectuent la partie la plus détestable du travail : récupérer ce peu d'informations, de descriptions, les digérer, les stocker et les régurgiter sous une forme acceptable lorsque cela est demandé.

Pour faciliter ce travail d'indexation du contenu, les plus grands moteurs de recherche se sont rassemblés et ont décidé d'un format commun, les sitemaps, pour que les webmasters puissent leur communiquer l'ensemble des pages disponibles sur un site, avec quelques autres métadonnées supplémentaires, leur facilitant *de facto* un peu le travail.

Pour le Web des données, on veut pouvoir agir de la même manière : en ayant une requête à formuler, on recherche l'ensemble de données qui est susceptible de convenir (on effectue une recherche sur un moteur de recherche sur certains mots-clés pour trouver des pages au contenu intéressant), on exploite ces informations (on lit l'article déniché).

Cependant, tout n'est pas si rose : alors que le Web des documents était visité par des êtres humains, capables de beaucoup de raisonnement, le Web des données est prévu pour être exploité par des machines, aux capacités cognitives plus faibles. Il ne suffit donc plus d'effectuer des recherches dans l'à-peu-près qui caractérise les moteurs actuels, il faut être capable de plus de précision, il faut que les ensembles de données soient décrits de manière plus fine. Les publicateurs doivent donc promouvoir leurs données aussi précisément que possible, avec des mots adaptés.

Bien que cette introduction puisse sembler accessible à tout public, la suite l'est nettement moins, il est nécessaire d'avoir bien compris les concepts de base du Web sémantique : triplet RDF, ontologie et vocabulaire, requête SPARQL, RDFa, etc. Si ces mots vous effraient, voir Introduction au Web sémantique (Lien 23) ou d'autres cours plus avancés (Lien 24). Les triplets RDF seront représentés en Turtle, donc compatible Notation3, les requêtes à l'aide de SPARQL.

## 3. Le vocabulaire VoID

*Vocabulary of Interlinked Datasets*.

Il s'agit de décrire le contenu des ensembles de données liées : cet ensemble contient 2042 liens de type foaf:depict vers tel autre ensemble, contenant principalement des données sur les avions de chasse. Tout ceci est décrit en RDF, ce qui permet d'effectuer des requêtes SPARQL sur une base pour chercher les meilleures sources de données pour son utilisation.

On a donc la première pièce du puzzle : on a l'équivalent d'un moteur de recherche, dont les données sont spécifiées précisément, on peut effectuer des requêtes, on ne se base pas sur des recherches en texte plein ou avec des approximations.

Ce vocabulaire est principalement constitué de deux classes : une pour décrire les ensembles de données (void:Dataset), l'autre pour indiquer les liens entre ces ensembles (void:Linkset).

Un void:Dataset est une collection de données publiées par un seul fournisseur, disponibles en RDF et accessibles (soit par des URI déréférençables, soit par un SPARQL endpoint, soit par une autre méthode).

Pour chaque lien, on utilise la classe void:Linkset, sous-classe de void:Dataset. Un tel triplet est toujours constitué d'un objet (une ressource de l'ensemble à décrire) et d'un sujet (une ressource d'un autre ensemble lié).

Par exemple, on peut décrire les relations entre DBpedia (Lien 25) et la base de données bibliographiques DBLP (Lien 26) :

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-
syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-
schema#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix void: <http://rdfs.org/ns/void#> .

:DBpedia rdf:type void:Dataset ;
         foaf:homepage <http://dbpedia.org/> .

:DBLP rdf:type void:Dataset ;
      foaf:homepage <http://www4.wiwiss.fu-
berlin.de/dblp/all> ;
      dcterms:subject
<http://dbpedia.org/resource/Computer_science> ;
      dcterms:subject
<http://dbpedia.org/resource/Journal> ;
      dcterms:subject
<http://dbpedia.org/resource/Proceedings> .

:DBpedia void:subset :DBpedia2DBLP  .

:DBpedia2DBLP rdf:type void:Linkset ;
              void:target :DBpedia ;
              void:target :DBLP .
```

En insérant ces triplets ainsi que bien d'autres décrivant des relations entre d'autres ensembles de données, on peut imaginer effectuer une requête pour déterminer quel ensemble regarder pour des données sur n'importe quel sujet :

```
SELECT DISTINCT ?dataset
WHERE {
  ?dataset a void:Dataset .
  ?dataset dcterms:subject
<http://dbpedia.org/resource/Journal> .
}
```

## 4. Écrire un fichier VoID

On va ici décrire la majorité des possibilités de ce vocabulaire pour un ensemble de données en développant un exemple basé sur DBpedia, mais avec beaucoup de parties fictives ne correspondant pas forcément à la réalité.

### 4.1. Description de l'ensemble de données

On commence par spécifier que l'on a affaire à un ensemble de données :

```
:DBpedia a void:Dataset .
```

#### 4.1.1. Informations de base

Cependant, cela n'apporte pas beaucoup d'informations. En réalité, beaucoup trop peu pour que l'on puisse en retirer quelque chose d'utile. On commence donc par préciser la page d'accueil (foaf:homepage), ainsi que d'autres pages utiles (foaf:page) :

```
:DBpedia a void:Dataset ;
    foaf:homepage <http://dbpedia.org/> ;
    foaf:page <http://ckan.net/package/dbpedia> ;
    foaf:page <http://dbpedia.org/Downloads> ;
    .
```

#### 4.1.2. Informations utiles

On peut ensuite ajouter quelques autres métadonnées : le titre (dcterms:title), une description (dcterms:description), les créateurs (dcterms:creator), éditeurs (dcterms:publisher) et contributeurs (dcterms:contributor), une description RDF de la source des données (dcterms:source), ainsi que les dates de création (dcterms:created), publication (dcterms:issued) et dernière modification (dcterms:modified).

Il est aussi probablement utile de fournir des informations de contact pour les créateurs, éditeurs et contributeurs à l'aide de foaf:mbox. On peut aussi préciser s'il s'agit d'une personne (foaf:Person) ou d'une organisation (foaf:Organization).

```
:DBpedia a void:Dataset ;
    dcterms:title "DBPedia" ;
    dcterms:description "RDF data extracted from
Wikipedia" ;
    dcterms:contributor :FU_Berlin ;
    dcterms:publisher :Alice ;
    dcterms:source
<http://dbpedia.org/resource/Wikipedia> ;
    dcterms:modified "2011-08-14"^^xsd:date ;
    .
:FU_Berlin a foaf:Organization ;
    rdfs:label "Freie Universität Berlin" ;
    foaf:homepage <http://www.fu-berlin.de/> ;
    .
:Alice a foaf:Person ;
    rdfs:label "Alice" ;
    foaf:mbox <mailto:alice@example.com> ;
    .
```

### 4.1.3. Licence

La licence est également un élément très important (Il vaut mieux utiliser une licence prévue pour des données qu'une licence générique, afin d'éviter de potentiels effets de bord et dégâts collatéraux. Notamment, Open Data Commons (Lien 27) en fournit quelques-unes, sur le principe de Creative Commons : un résumé « humainement lisible » de la licence en plus du texte légal.) que l'on peut spécifier avec dcterms:license. Il est également possible à l'éditeur de renoncer à une série de droits sur ces données pour en favoriser l'utilisation, on utilise alors waiver:norms.

```
:DBpedia a void:Dataset ;
    dcterms:title "DBPedia" ;
    dcterms:description "RDF data extracted from
Wikipedia" ;
    dcterms:contributor :FU_Berlin ;
    dcterms:publisher :Alice ;
    dcterms:source
<http://dbpedia.org/resource/Wikipedia> ;
    dcterms:modified "2011-08-14"^^xsd:date ;
    dcterms:license
<http://www.opendatacommons.org/odc-public-
domain-dedication-and-licence/> ;
    wv:norms
<http://www.opendatacommons.org/norms/odc-by-sa/>
;
    wv:waiver """To the extent possible under
law, The Example Organisation has waived all
          copyright and related or neighboring
rights to The Example Dataset.""" ;
    .
:FU_Berlin a foaf:Organization ;
    rdfs:label "Freie Universität Berlin" ;
    foaf:homepage <http://www.fu-berlin.de/> ;
    .
:Alice a foaf:Person ;
    rdfs:label "Alice" ;
    foaf:mbox <mailto:alice@example.com> ;
    .
```

### 4.1.4. Sujets

Toute la substance de ce vocabulaire, par contre, tient dans les sujets que traite l'ensemble, ce sans quoi le reste n'a pas grande utilité. On utilise le prédicat dcterms:subject avec un sujet, de préférence une URI de ressource sur DBpedia (afin d'uniformiser les sujets et de faciliter les recherches).

```
:DBpedia a void:Dataset ;
    dcterms:title "DBPedia" ;
    dcterms:description "RDF data extracted from
Wikipedia" ;
    dcterms:contributor :FU_Berlin ;
    dcterms:publisher :Alice ;
    dcterms:source
<http://dbpedia.org/resource/Wikipedia> ;
    dcterms:modified "2011-08-14"^^xsd:date ;
    dcterms:license
<http://www.opendatacommons.org/odc-public-
domain-dedication-and-licence/> ;
    wv:norms
<http://www.opendatacommons.org/norms/odc-by-sa/>
;
    wv:waiver """To the extent possible under
law, The Example Organisation has waived all
          copyright and related or neighboring
rights to The Example Dataset.""" ;
        dcterms:subject
<http://dbpedia.org/resource/Encyclopedia> ;
    .
:FU_Berlin a foaf:Organization ;
    rdfs:label "Freie Universität Berlin" ;
    foaf:homepage <http://www.fu-berlin.de/> ;
    .
:Alice a foaf:Person ;
    rdfs:label "Alice" ;
    foaf:mbox <mailto:alice@example.com> ;
   .
```

DBpedia ne contient cependant pas des sujets pour tous les ensembles de données, particulièrement s'il concerne la génétique ou un autre sujet peu répandu. Dans ce cas, on utilise des URI largement répandues dans les communautés concernées.

### 4.1.5. Fonctionnalités techniques

On peut spécifier une série de fonctionnalités techniques, comme les possibilités de sérialisation des triplets RDF, à l'aide de void:feature. Au besoin, on peut définir ses propres fonctionnalités.

```
:DBpedia a void:Dataset ;
    dcterms:title "DBPedia" ;
    dcterms:description "RDF data extracted from
Wikipedia" ;
    dcterms:contributor :FU_Berlin ;
    dcterms:publisher :Alice ;
    dcterms:source
<http://dbpedia.org/resource/Wikipedia> ;
    dcterms:modified "2011-08-14"^^xsd:date ;
    dcterms:license
<http://www.opendatacommons.org/odc-public-
domain-dedication-and-licence/> ;
    wv:norms
<http://www.opendatacommons.org/norms/odc-by-sa/>
;
    wv:waiver """To the extent possible under
law, The Example Organisation has waived all
          copyright and related or neighboring
rights to The Example Dataset.""" ;
        dcterms:subject
<http://dbpedia.org/resource/Encyclopedia> ;
        void:feature
<http://www.w3.org/ns/formats/N3> ;
        void:feature
<http://www.w3.org/ns/formats/N-Triples> ;
        void:feature
<http://www.w3.org/ns/formats/RDF_XML> ;
```

```
        void:feature
<http://www.w3.org/ns/formats/RDFa> ;
        void:feature
<http://www.w3.org/ns/formats/Turtle> ;
        void:feature
<http://www.w3.org/ns/formats/HTTPCachingETags> ;
    .
:FU_Berlin a foaf:Organization ;
    rdfs:label "Freie Universität Berlin" ;
    foaf:homepage <http://www.fu-berlin.de/> ;
    .
:Alice a foaf:Person ;
    rdfs:label "Alice" ;
    foaf:mbox <mailto:alice@example.com> ;
    .
:HTTPCachingETags a void:TechnicalFeature;
    rdfs:label "HTTP ETag support" ;
    rdfs:comment "The dataset supports HTTP
caching using ETags" ;
    rdfs:seeAlso
<http://www.w3.org/Protocols/rfc2616/rfc2616-
sec14.html#> ;
    .
```

#### 4.1.6. Sous-ensembles

On peut simplement définir un ensemble comme plusieurs sous-ensembles, on peut alors définir des propriétés ne se rapportant qu'à un seul sous-ensemble et pas au reste.

```
:DBpedia a void:Dataset;
    void:subset :DBpedia_shortabstracts;
    void:subset :DBpedia_infoboxes;
    .
:DBpedia_shortabstracts a void:Dataset;
    dcterms:title "DBpedia Short Abstracts";
    dcterms:description "Short Abstracts (max.
500 chars long) of Wikipedia Articles";
    void:dataDump
<http://downloads.dbpedia.org/3.3/en/shortabstrac
t_en.nt.bz2>;
    .
:DBpedia_infoboxes a void:Dataset;
    dcterms:title "DBpedia Infoboxes";
    dcterms:description "Information that has
been extracted from Wikipedia infoboxes.";
    void:dataDump
<http://downloads.dbpedia.org/3.3/en/infobox_en.n
t.bz2>;
    .
```

void:dataDump sera introduit très prochainement, il s'agit de lier un fichier contenant tous les triplets de l'ensemble sérialisés.

#### 4.1.7. Statistiques

On peut définir un certain nombre de statistiques pour un ensemble de données : le nombre de triplets (void:triples), d'entités (Une entité est une ressource qui a une URI devant correspondre à la void:uriRegexPattern de l'ensemble, si définie. Chaque auteur de fichier VoID peut aussi poser d'autres conditions pour définir une entité.) (void:entities), de classes (void:classes), de propriétés (void:properties), de sujets distincts (void:distinctSubjects), d'objets distincts (void:distinctObjects) et de documents (On considère qu'un ensemble est constitué de documents quand il est disponible sous forme de fichiers RDF ou de pages Web annotées à l'aide de RDFa. On ne compte généralement pas les pages Web sans annotations ni les images. Cette propriété est prévue pour les ensembles où il n'est pas aisé de déterminer le nombre total de triplets ou d'entités.) (void:documents).

```
:DBpedia a void:Dataset ;
    dcterms:title "DBPedia" ;
    dcterms:description "RDF data extracted from
Wikipedia" ;
    dcterms:contributor :FU_Berlin ;
    dcterms:publisher :Alice ;
    dcterms:source
<http://dbpedia.org/resource/Wikipedia> ;
    dcterms:modified "2011-08-14"^^xsd:date ;
    dcterms:license
<http://www.opendatacommons.org/odc-public-
domain-dedication-and-licence/> ;
    wv:norms
<http://www.opendatacommons.org/norms/odc-by-sa/>
;
    wv:waiver """To the extent possible under
law, The Example Organisation has waived all
          copyright and related or neighboring
rights to The Example Dataset.""" ;
        dcterms:subject
<http://dbpedia.org/resource/Encyclopedia> ;
        void:feature
<http://www.w3.org/ns/formats/N3> ;
        void:feature
<http://www.w3.org/ns/formats/N-Triples> ;
        void:feature
<http://www.w3.org/ns/formats/RDF_XML> ;
        void:feature
<http://www.w3.org/ns/formats/RDFa> ;
        void:feature
<http://www.w3.org/ns/formats/Turtle> ;
        void:feature
<http://www.w3.org/ns/formats/HTTPCachingETags> ;
        void:triples 1000000000;
    void:entities 4200000;
    .
:FU_Berlin a foaf:Organization ;
    rdfs:label "Freie Universität Berlin" ;
    foaf:homepage <http://www.fu-berlin.de/> ;
    .
:Alice a foaf:Person ;
    rdfs:label "Alice" ;
    foaf:mbox <mailto:alice@example.com> ;
    .
:HTTPCachingETags a void:TechnicalFeature;
    rdfs:label "HTTP ETag support" ;
    rdfs:comment "The dataset supports HTTP
caching using ETags" ;
    rdfs:seeAlso
<http://www.w3.org/Protocols/rfc2616/rfc2616-
sec14.html#> ;
    .
```

Ces statistiques permettent de quantifier la qualité d'un ensemble de données, notamment en regardant le nombre de triplets disponibles en moyenne pour chaque entité, le nombre de sujets et d'objets distincts montrant la diversité de l'ensemble.

### 4.2. Accès aux triplets

Jusqu'à présent, on n'a pas donné de manière d'accéder aux triplets. On doit donc effectuer une requête, aller sur le site Web... puis chercher l'accès aux triplets. Rien de vraiment automatisable. Sauf si on indique précisément comment dénicher ces triplets !

On simplifiera fortement l'exemple précédent, le réduisant à un nombre de triplets proche de zéro.

#### 4.2.1. SPARQL endpoint

On peut indiquer un habituel SPARQL endpoint :

```
:DBpedia a void:Dataset ;
    void:sparqlEndpoint
<http://dbpedia.org/sparql>;
    .
```

#### 4.2.2. Fichiers RDF

Il ne serait pas étonnant d'aussi trouver des fichiers RDF contenant l'ensemble des triplets disponibles dans la base à un moment donné. On peut préciser plusieurs liens vers des fichiers contenant des triplets sérialisés dans un format habituel (sans qu'il y ait moyen de le préciser explicitement, on peut se baser sur les fonctionnalités techniques (Ce genre d'informations est généralement disponible cependant sur la page des téléchargements. Pour Dbpedia (Lien 28), il est précisé explicitement que les triplets sont disponibles en N-Triples et N-Quads compressés en BZip2.)), possiblement compresser. On doit lier directement les fichiers, pas une page permettant de les télécharger.

```
:DBpedia a void:Dataset ;
    void:sparqlEndpoint
<http://dbpedia.org/sparql>;
    void:dataDump
<http://downloads.dbpedia.org/3.6/en/article_cate
gories_en.nt.bz2> ;
    void:dataDump
<http://downloads.dbpedia.org/3.6/el/disambiguati
ons_el.nq.bz2> ;
    .
```

#### 4.2.3. OpenSearch

OpenSearch est un format de document permettant de décrire un moteur de recherche (notamment, les navigateurs Web qui le supportent proposent d'ajouter un moteur de recherche disposant d'un fichier OpenSearch à la liste des moteurs de recherche disponibles dans les barres d'outils appropriées). Il s'agit ici d'une recherche textuelle classique.

```
:DBpedia a void:Dataset ;
    void:sparqlEndpoint
<http://dbpedia.org/sparql>;
    void:dataDump
<http://downloads.dbpedia.org/3.6/en/article_cate
gories_en.nt.bz2> ;
    void:dataDump
<http://downloads.dbpedia.org/3.6/el/disambiguati
ons_el.nq.bz2> ;
    void:openSearchDescription
<http://www.sindice.com/opensearch.xml> ;
    .
```

#### 4.2.4. URI de recherche

On dispose parfois aussi d'URI permettant d'effectuer des recherches, donnant la description RDF de l'entité passée en paramètre. Par exemple, on peut effectuer une recherche sur Cat via Sindice à l'URI suivante : Lien 29. On décrira cette manière de rechercher comme suit :

```
:DBpedia a void:Dataset ;
    void:sparqlEndpoint
<http://dbpedia.org/sparql>;
    void:dataDump
<http://downloads.dbpedia.org/3.6/en/article_cate
gories_en.nt.bz2> ;
    void:dataDump
<http://downloads.dbpedia.org/3.6/el/disambiguati
ons_el.nq.bz2> ;
    void:openSearchDescription
<http://www.sindice.com/opensearch.xml> ;
    void:uriLookupEndpoint
<http://api.sindice.com/v2/search?qt=term&q=> ;
    .
```

### 4.3. Description des liens

On crée simplement un lien entre deux ensembles décrits dans le même fichier comme ceci :

```
:DBpedia_Geonames a void:Linkset;
    void:target :DBpedia ;
    void:target :Geonames ;
    .
```

On ne peut présenter ainsi que deux ensembles à la fois, chacun avec une propriété void:target.

#### 4.3.1. Statistiques

On peut définir le nombre de triplets concernés par ce lien (ou toute autre statistique possible, étant donné que la classe void:linkset dérive de void:Dataset). Cela est néanmoins plus utile lorsque l'on précise le prédicat sur lequel porte le lien.

```
:DBpedia_Geonames a void:Linkset;
    void:target :DBpedia ;
    void:target :Geonames ;
    void:triples 242000 ;
    .
```

#### 4.3.2. Prédicat visé par le lien

La propriété void:linkPredicate sert à préciser la nature du lien entre les deux ensembles.

```
:DBpedia_Geonames a void:Linkset;
    void:target :DBpedia ;
    void:target :Geonames ;
    void:linkPredicate owl:sameAs ;
    .
```

### 4.4. Utilisation pratique du fichier

C'est bien de créer toutes ces données, encore faut-il qu'elles soient exploitables par des moteurs de recherche tels que Sindice. Tout d'abord, il faut mettre ce fichier à proximité de l'ensemble de données. Plusieurs options : un fichier void.ttl au format Turtle à la racine du site, servir les formats HTML et RDF pour l'URI racine du site (voir Cool URIs : Lien 30) ou intégrer le tout dans la page grâce à RDFa.

Il n'est pas recommandé d'intégrer ces données à un fichier de triplets RDF contenant toutes les données. Par contre, on peut ajouter un triplet pointant vers cette description :

```
<document.rdf> void:inDataset
<void.ttl#MyDataset> .
```

Pour soumission à Sindice, il faut se rendre dans l'onglet Submit (Lien 31) du site et ajouter l'URI du fichier VoID dans la grande boîte. (On retournera bientôt sur cette page pour ajouter un sitemap, juste en dessous.)

Sur le VoID store de RKB (Lien 32), la procédure est fortement similaire, ainsi que sur le VoID browser de Talis (Lien 33) ou Ping the Semantic Web (Lien 34). Comme sur le Web des documents, cela ne fait pas de mal de soumettre ce fichier à divers endroits pour maximiser sa visibilité.

## 5. Les sitemaps

Deuxième brique identifiée dans l'introduction, les sitemaps, des documents listant toutes les pages d'un site avec d'autres informations potentiellement utiles au moteur de recherche.

Au contraire, sur le Web des données, comme on peut effectuer des requêtes SPARQL pour savoir tout ce qui est disponible, il n'est pas si utile de lister quelque part tout le contenu ; par contre, préciser quelles sont les méthodes possibles pour récupérer les triplets peut être utile : si on veut effectuer énormément de requêtes sur un ensemble, autant télécharger les triplets le composant et effectuer le traitement en local, pour éviter de surcharger le serveur distant. *A contrario*, pour deux ou trois requêtes, autant bénéficier du SPARQL endpoint ou de l'accès par URI. (Il faut évidemment aussi prendre en compte la taille des données à télécharger : on ne téléchargera pas l'intégralité de DBpedia pour un malheureux millier de requêtes. Par contre, pour des bases de cette taille, il est souvent possible de ne télécharger qu'une partie des triplets, ce qui ne représentera qu'une infime partie de la centaine de gigaoctets de données disponibles.)

Mais peut-on le faire ? Comment peut-on s'assurer qu'on aura *exactement* les mêmes données d'un côté ou de l'autre ? C'est le but des sitemaps sémantiques.

### 5.1. Prévenir de l'existence d'un sitemap

Tout d'abord, il faut que l'on puisse trouver ce sitemap en visitant le site. On peut l'indiquer grâce au fichier robots.txt.

```
Sitemap: http://www.site.com/sitemap.xml
```

### 5.2. Écrire le sitemap

Il ne s'agit que d'une extension du format de sitemaps déjà largement admis. Si le site en dispose déjà, il suffira d'y ajouter quelques lignes. On va partir d'un exemple complet à détailler.

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset
xmlns="http://www.sitemaps.org/schemas/sitemap/0.
9"
        xmlns:sc="http://sw.deri.org/2007/07/site
mapextension/scschema.xsd">
  <sc:dataset>
    <sc:datasetLabel>Example Corp. Product
Catalog</sc:datasetLabel>
    <sc:datasetURI>http://example.com/catalog.rdf
#catalog</sc:datasetURI>
    <sc:sampleURI>http://example.com/products/wid
gets/X42</sc:sampleURI>
    <sc:sampleURI>http://example.com/products/cat
egories/all</sc:sampleURI>
    <sc:sparqlEndpointLocation slicing="subject-
object">http://example.com/sparql</sc:sparqlEndpo
intLocation>
    <sc:dataDumpLocation>http://example.com/data/
catalogdump.rdf.gz</sc:dataDumpLocation>
    <sc:dataDumpLocation>http://example.org/data/
catalog_archive.rdf.gz</sc:dataDumpLocation>
    <sc:dataDumpLocation>http://example.org/data/
product_categories.rdf.gz</sc:dataDumpLocation>
    <changefreq>weekly</changefreq>
  </sc:dataset>
</urlset>
```

#### 5.2.1. Ensemble de données

```
  <sc:dataset>
    <!-- ... -->
  </sc:dataset>
```

On définit d'abord un ensemble de données. Un sitemap peut définir plusieurs ensembles de données.

#### 5.2.2. Informations générales

```
<sc:datasetLabel>Example Corp. Product
Catalog</sc:datasetLabel>
<sc:datasetURI>http://example.com/catalog.rdf#cat
alog</sc:datasetURI>
```

Ces deux éléments sont optionnels et fournissent des métadonnées sur l'ensemble de données : respectivement, le nom et l'URI. Généralement, la résolution de l'URI donne plus d'informations sur l'ensemble, possiblement en RDF, mais ce n'est pas requis.

#### 5.2.3. URI d'exemples

```
<sc:sampleURI>http://example.com/products/widgets
/X42</sc:sampleURI>
<sc:sampleURI>http://example.com/products/categor
ies/all</sc:sampleURI>
```

Ces URI optionnelles peuvent pointer vers des échantillons représentatifs de l'ensemble. Elles servent comme point de base pour une exploration humaine de l'ensemble.

#### 5.2.4. Triplets

```
<sc:sparqlEndpointLocation slicing="subject-
object">http://example.com/sparql</sc:sparqlEndpo
intLocation>
<sc:dataDumpLocation>http://example.com/data/cata
logdump.rdf.gz</sc:dataDumpLocation>
<sc:dataDumpLocation>http://example.org/data/cata
log_archive.rdf.gz</sc:dataDumpLocation>
<sc:dataDumpLocation>http://example.org/data/prod
uct_categories.rdf.gz</sc:dataDumpLocation>
```

Le point central (obligatoire), les sources de données, l'objectif même des sitemaps. On a défini ici un SPARQL endpoint (il peut y en avoir *au plus* un par ensemble). On a ajouté quelques fichiers RDF, dont l'union est dite être l'ensemble des données. Cette union doit d'ailleurs être

disponible par le biais du SPARQL endpoint.

### 5.2.5. Modifications

```
<changefreq>weekly</changefreq>
```

On spécifie ici la fréquence de mise à jour de l'ensemble, cette balise optionnelle est héritée du protocole sitemap. On a aussi la balise <lastmod> pour indiquer la dernière mise à jour (au format W3C : AAAA-MM-DD ou AAAA-MM-DD*T*HH:MM:SS+HH:MM.

### 5.3. Référencer le sitemap sur Sindice

Mettre à disposition le sitemap par robots.txt ne sera pas suffisant pour que les moteurs de recherche y aient accès, tout comme pour le Web des documents.

On va ici le référencer sur Sindice, principal moteur de recherche pour le Web sémantique. On retourne à l'onglet Submit (Lien 31), mais pour remplir le deuxième formulaire : il suffit d'y préciser l'URL du sitemap puis de cliquer sur le bouton. C'est simple et pourtant le sitemap est envoyé et sera référencé.

Il faut aussi avoir envoyé son fichier VoID pour être référençable sur Sindice.

## 6. Conclusions

Référencer ses données dans le monde du Web sémantique n'est pas chose compliquée pour qui a déjà une petite expérience des technologies mises en place. On peut voir d'ailleurs qu'il s'agit d'une adaptation des solutions déjà trouvées pour le Web des documents : pourquoi réinventer la roue ? Surtout que l'on dispose déjà d'outils qui fonctionnent très bien (les sitemaps, mais aussi les SPARQL endpoints, Turtle et bien d'autres) !

*Retrouvez l'article de Thibaut Cuvelier en ligne : Lien 35*

# Le guide ultime des microformats : références et exemples

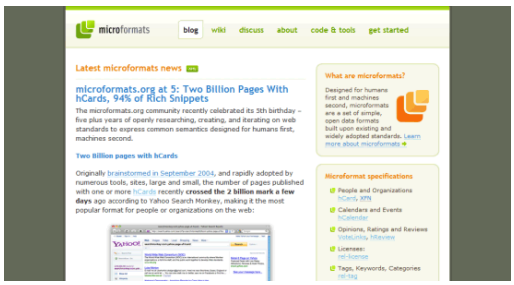
Si vous n'êtes pas familier du concept POSH (Plain Old Semantic HTML), la première chose à savoir c'est que produire du code sémantique, qui reflète la valeur du contenu textuel (en plus de la mise en forme), est une composante critique du processus de conception Web.
Alors que le HTML dispose d'une kyrielle d'éléments par lesquels le contenu prend du sens, une foultitude de microformats (conventions) ont été créés pour mieux représenter les données qui composent votre page.

## 1. Les microformats, c'est quoi ?

Bien qu'ils ne fassent pas partie des spécifications HTML du W3C, ils offrent un assortiment utile de conventions de nommage (en utilisant les attributs class, id, rel et rev) qui identifient les points d'intérêt sur une page. Ils permettent de mettre en avant du contenu, tel que les évènements de calendrier, un lien vers l'acceptation de vos licences (dont la GPL) et même des choses plus légères telles que les recettes de cuisine.

Si les microformats ne font pas encore partie du standard W3C, les navigateurs Web n'ont pas attendu pour en assurer le support.

Les microformats valent vraiment la peine d'y porter un intérêt et de les implémenter dans les sites que vous réalisez.

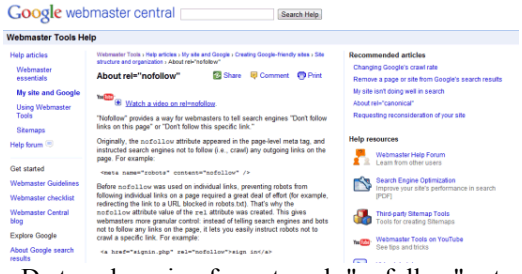



Le site officiel des microformats dispose d'un wiki et d'un forum et met différents outils à votre disposition.

Vous utilisez peut-être déjà les microformats si vous utilisez un CMS tel que WordPress, car ce dernier supporte nativement des formats de données simples, comme l'attribut *rel*.

Si vous êtes nouveau dans l'univers des microformats, vous vous demandez certainement **pourquoi vous devriez vous embêter à les utiliser.**

Il y a certes des avantages et des inconvénients, mais tout ce qui aidera votre site à être mieux compris par les robots (*spiders, crawlers*) qui indexent vos pages vaut bien cet effort supplémentaire, non ?




De tous les microformats, rel="nofollow" est probablement le plus connu.

Parce que les microformats s'appuient sur la syntaxe et les attributs conventionnels du HTML, vous pouvez les utiliser aussi en XHTML. Même les pages XML (comme les flux RSS ou ATOM) peuvent profiter des microformats. Cela accroit ainsi leur utilisation potentielle. Ils se combinent également très bien avec le RDFa et les autres métadonnées.

### 1.1. Les avantages des microformats

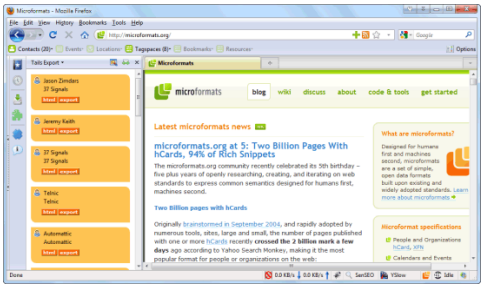
1. Ils améliorent la valeur sémantique de votre contenu.
2. D'autres applications Web peuvent les utiliser pour découvrir les contenus de votre site. Ils

peuvent également servir d'interface avec vos données.
3. Les réseaux sociaux les implémentent dans les profils utilisateurs pour que des services tiers puissent interagir avec eux.
4. Des extensions de navigateurs existent pour donner aux utilisateurs l'accès aux données contenues dans les microformats. Par exemple, Michromeformats est une extension Google Chrome qui découvre pour vous les microformats embarqués dans une page web.
5. Les robots (comme Googlebot) les utilisent pour l'indexation des sites.

### 1.2. Les inconvénients des microformats

1. Ils nécessitent davantage de code HTML.
2. Ils sont encore une chose que vous devez apprendre et maintenir.
3. Ils existent pour relativement peu de types de données.
4. Ils attirent l'attention sur vos contenus (qui peuvent être data minés).
5. Les navigateurs ne les supportent pas uniformément.

L'extension Operator pour Firefox détecte les microformats et les rend lisibles.



Le microformat hCard permet aux extensions Firefox telles que Tails Export de découvrir et de s'interfacer avec la carte de visite virtuelle de quelqu'un.

## 2. Tableau de référence des microformats

Chaque microformat a un but unique de présenter un certain type d'informations et tous sont potentiellement utiles en fonction de vos besoins.

Bien que vous pourrez trouver des informations exhaustives sur le site des microformats, voici un récapitulatif rapide de ce qu'on y trouve :

| Nom | But |
|---|---|
| ADR : Lien 36 | Marquer une adresse postale. |
| Geo : Lien 37 | Marquer une position géographique. |
| hAtom : Lien 38 | Ajouter du contenu syndicable. |
| hAudio : Lien 39 | Décrire un podcast ou un fichier audio. |
| hCalendar : Lien 40 | Marquer un événement ou lister du contenu par date (concerts, spectacles…). |
| hCard : Lien 41 | Exploiter les contacts d'affaires ou personnels (carte de visite). |
| hListing : Lien 42 | Lister des biens et des services. |
| hMedia : Lien 43 | Lister des médias. |
| hNews : Lien 44 | Utilise hAtom pour les nouvelles journalistiques. |
| hProduct : Lien 45 | Embarquer des informations supplémentaires sur un produit. |
| hRecipe : Lien 46 | Marquer des recettes de cuisine. |
| hResume : Lien 47 | Présenter les expériences d'un CV. |
| hReview : Lien 48 | Avis et notes de produits et services. |
| rel : Lien 49 | L'attribut rel est un microformat pour les éléments HTML, quelques exemples populaires : rel="license" (Lien 50), rel="nofollow" (Lien 51), rel="tag" (Lien 52), rel="directory" (Lien 53), rel="enclosure" (Lien 54), rel="home" (Lien 55), rel="payment" (Lien 56). |
| Robot Exclusion Profile : Lien 57 | Donner des instructions aux robots. |
| VoteLinks : Lien 58 | Fournir des options pour aimer ou ne plus aimer un lien. |
| XFN : Lien 59 | Décrire une relation avec un site Web. |
| XFolk : Lien 60 | Lister des liens favoris. |
| XMDP : Lien 61 | Ajouter des ressources sur le profil d'une page. |
| XOXO : Lien 62 | Mettre en avant un document ou une liste d'objets. |

### 2.1. Les valeurs de l'attribut rel

Pour compléter le tableau ci-dessus, voici une description des valeurs de l'attribut rel (raccourci de *relationship*) :

- **license** : identifie les accords de licences (tel que Creative Commons ou GPL) sur une page ;
- **nofollow** : indique aux moteurs de recherche de ne pas ajouter de poids ou de valeur à la ressource liée ;
- **tag** : applique des mots-clés aux liens pour construire un nuage de tags ou de catégories ;
- **directory** : indique une liste dans un répertoire (comme un dossier) sur le site courant ;
- **enclosure** : pour les liens qui pointent vers des fichiers téléchargeables ;
- **home** : fournit un lien permanent vers la page d'accueil d'un site ;
- **payment** : indique si un lien pointe vers une page de paiement ou d'achat.

## 3. Utiliser les microformats : exemples

Recommander d'utiliser les microformats, c'est bien, mais

fournir des exemples, c'est mieux ! Nous allons voir ensemble quelques exemples pour chaque microformat qui peut être implémenté sur votre site.

Pour commencer, le concept-clé à comprendre : un microformat s'identifie par un bout de donnée contenu dans l'attribut de class ou id d'une balise HTML.

La balise peut jouer un rôle dans le type de donnée à afficher (comme les liens), mais si aucune alternative sémantique n'existe, vous pouvez utiliser un div ou un span autour du contenu en question. Même si l'utilisation d'un span ne semble pas très élégante, il ajoute une signification spéciale dans ce cas.

### 3.1. Adr

```
<ul class="adr">

  <li class="street-address">123 North
Street</li>
  <li class="locality">Manchester</li>
  <li class="postal-code">MX43 991</li>

  <li class="country-name">UK</li>
</ul>
```

**Racine** : adr.
Valeurs d'attributs :

- post-office-box ;
- extended-address ;
- street-address ;
- locality ;
- region ;
- postal-code ;
- country-name.

### 3.2. Geo

```
<p class="geo">
  <abbr class="latitude" title="37.408183">N 37°
24.491</abbr> -
  <abbr class="longitude" title="-122.13855">W
122° 08.313</abbr>

</p>
```

**Racine** : geo.
Valeurs d'attributs obligatoires :

- latitude ;
- longitude.

### 3.3. hAtom

```
<div class="hAtom">
  <div class="hentry">
    <h3 class="entry-title">I Love
Microformats</h3>
    <abbr class="published" title="2010-08-
28T13:14:37-07:00">Aug 28, 2010</abbr>

    <p class="category"><a href="/category/rdf"
rel="tag">RDF</a></p>
    <p><a href="#" title="Post a comment">What do
you think of this post?</a></p>

    <div class="entry-content">
      <p>Place your content right here for
maximum impact!</p>
    </div>
    <dl>
      <dt>Tags:</dt>

      <dd><a href="/tag/standards/"
rel="tag">standards</a></dd>
      <dd><a href=" /tag/microformats/"
rel="tag">microformats</a></dd>

    </dl>
  </div>
</div>
```

**Racine** : hAtom, hFeed.
Valeur d'attribut :

- hentry ;
- entry-title ;
- entry-content ;
- entry-summary ;
- bookmark ;
- published ;
- updated ;
- author.

### 3.4. hAudio

```
<p class="haudio">
  <em class="fn">Bohemian Rhapsody</em>
  by <span class="contributor vcard">
  <em class="fn org">Queen</em></span>

  found on <em class="album">A Night at the
Opera</em>
</p>
```

**Racine** : hAudio.
Valeurs d'attributs obligatoires :

- fn ;
- album.

Valeurs d'attributs optionnels :

- contributor ;
- duration ;
- item ;
- position ;
- category ;
- published ;
- photo ;
- description ;
- sample ;
- enclosure ;
- payment ;
- price (currency, amount).

### 3.5. hCalendar

Vous pouvez utiliser le générateur hCalendar plutôt que d'écrire le code à la main : Lien 63.

```
<p class="vEvent">
  <a class="url"
href="http://www.yoursitehere.com/">MySite</a>

  <span class="summary">New website
launch</span>:
  <abbr class="dtstart" title="20091202">December
```

```
2</abbr>-
  <abbr class="dtend" title="20091204">4</abbr>,
at
  <span class="location">Google College, London,
UK</span>

</p>
```

**Racine** : vCalendar, vEvent.
Valeurs d'attributs obligatoires :

- dtstart ;
- summary.

Valeurs d'attributs obligatoires :

- location ;
- url ;
- dtend ;
- duration ;
- rdate ;
- rrule ;
- category ;
- description ;
- uid ;
- geo (latitude, longitude) ;
- attendee (partstat, role) ;
- contact ;
- organizer ;
- attach ;
- status.

### 3.6. hCard

Vous pouvez aussi utiliser le générateur hCard plutôt que d'écrire le code à la main : Lien 64.

```
<ul id="hCard-John-Doe" class="vcard">
  <li class="fn">John Doe</li>
  <li class="org">Special Stores</li>

  <li><a class="email"
href="mailto:John@doe.org">John@doe.org</a></li>
  <li class="adr">
    <ul>

      <li class="street-address">44 Semantic
Drive</li>,
      <li class="locality">Markup City</li>,
      <li class="region">World Wide Web</li>,
      <li class="postal-code">BP33 9HQ</li>

      <li class="country-name">Internet</li>
    </ul>
  </li>
  <li class="tel">01234 56789</li>

 </ul>
```

**Racine** : hCard.
Valeurs d'attributs obligatoires :

- fn ;
- n (family-name, given-name, additional-name, honorific-prefix, honorific-suffix).

Valeurs d'attributs optionnels :

- adr (post-office-box, extended-address, street-address, locality, region, postal-code, country-name, type, value) ;
- agent ;
- bday ;
- category ;
- class ;
- email (type, value) ;
- geo (latitude, longitude) ;
- key ;
- label ;
- logo ;
- mailer ;
- nickname ;
- note ;
- org (organization-name, organization-unit) ;
- photo ;
- rev ;
- role ;
- sort-string ;
- sound ;
- tel (type, value) ;
- title ;
- tz ;
- uid ;
- url.

### 3.7. hListing

```
<div class="hlisting">
  <p>
    <span class="item fn">Office space</span>
    <span class="offer rent">to rent</span>(<abbr
class="dtlisted" title="20100202">2/2/10</abbr>)
  </p>

  <p class="description">50-square-foot space
available in local tech office at:
    <div class="location adr">
      <span class="street-address">123 Microland
Road.</span>
      <span class="locality">Cyberspace</span>,
<span class="region">XD</span>

      <span class="postal-code">12345</span>
      <span class="country">Mars</span>
    </div>
    Available during <abbr class="dtexpired"
title="20100401">April 2010</abbr>

    for <span class="price">$1500/qtr</span>
  </p>
  <div class="lister vcard">
    Contact:
    <span class="fn">John Doe</span>

 at <span class="tel"><span class="value">(01)
12345-678900</span>(<abbr class="type"
title="cell">C</abbr>)</span>

  </div>
</div>
```

**Racine** : hlisting.
Valeurs d'attributs obligatoires :

- description ;
- lister (fn, email, url, tel) ;
- action (sell, rent, trade, meet, announce, offer, wanted, event, service).

Valeurs d'attributs optionnels :

- version ;
- dtlisted ;
- dtexpired ;
- price ;
- item (fn, url, photo, geo, adr) ;
- summary ;
- tag ;
- permalink.

### 3.8. hMedia

```
<div class="hmedia">
  <h3 class="fn">Introduction to the Open Media
Web</h3>
  <object class="player" type="application/x-
shockwave-flash"
data="http://www.exampleurl.com/video.swf">

    <param name="movie"
value="http://www.exampleurl.com/video.swf"/>
    <param name="allowScriptAccess"
value="always"/>
    <param name="allowFullScreen" value="true"/>

  </object>
  <ul>
    <li><a rel="enclosure" type="video/mp4"
title="Download the movie"
href="http://www.exampleurl.com/video.mp4">Video.
mp4</a></li>
  </ul>
</div>
```

**Racine** : hMedia.
Valeur d'attribut :

- fn ;
- contributor ;
- photo ;
- player ;
- enclosure.

### 3.9. hNews

```
<div class="hnews hentry item">
  <h4 class="entry-title">Microformats are
awesome</h4>
  <p class="author vcard">

  By <span class="fn" >John Doe</span>,
  <span class="source-org vcard dateline"><span
class="org fn">Associated Press</span></span> -
  <span class="updated" title="2010-04-19">19
April 2010</p>

  <p>News story</p>
</div>
```

**Racine** : hNews.
Valeur d'attribut obligatoire :

- hentry ;
- item ;
- entry-title ;
- author ;
- source-org ;
- vcard ;
- updated.

Valeurs d'attributs optionnels :

- dateline ;
- geo (latitude, longitude) ;
- item-license ;
- principles.

### 3.10. hProduct

```
<ul class="hproduct">
  <li class="brand">MySite!</li>
  <li class="category">Software</li>

  <li class="fn">Microsoft Office 2007</li>
  <li class="description">The world's most
popular office suite.</li>
  <li
class="url">http://office.microsoft.com</li>

</ul>
```

**Racine** : hProduct.
Valeur d'attribut obligatoire :

- fn.

Valeurs d'attributs optionnels :

- brand ;
- category ;
- price ;
- description ;
- photo ;
- url ;
- review ;
- listing ;
- identifier (type (model, mpn, upc, isbn, issn, ean, jan, sn, vin, sku), value).

### 3.11. hRecipe

```
<div class="hrecipe">
<h3 class="fn">Quick noodles</h3>
  <p class="summary">Noodles are quick and easy,
like this example!</p>
  <p class="ingredient hcard"><span
class="value">2.5</span><span
class="type">kilogram</span>bag of instant
noodles.</p>

  <ul class="instructions">
    <li>Put water on to boil,</li>
    <li>Add the powder for the sauce,</li>
    <li>Add the noodles, and stir till
ready.</li>

  </ul>
  <p>Enough for <span class="yield">1
adult</span>.</p>
  <p>Preparation time is approximately <span
class="duration">5 <abbr
title="minutes">mins</abbr></span>.</p>

  <p class="nutrition hcard">Noodles have more
than <span class="value">500</span> <span
class="type">joules</span> of energy.</p>

</div>
```

**Racine** : hRecipe.
Valeurs d'attribut obligatoires :

- fn ;
- ingredient (value, type).

Valeurs d'attributs optionnels :

- yield ;
- instructions ;
- duration ;
- photo ;
- summary ;
- author ;
- published ;
- nutrition (value, type), tag.

### 3.12. hResume

Vous pouvez aussi utiliser le générateur hResume plutôt que d'écrire le code à la main : Lien 65.

```
<div id="hResume">
  <p class="summary">I have been producing
microformatted data for years</p>
  <ul class="vcard">

    <li class="fn">Jane Doe</li>
    <li class="adr">
      <span class="street-address">44 Broadband
Street</span>
      <span class="locality">Microland</span>,
<span class="region">Internet</span>

      <span class="postal-code">QW11
ER4</span></li>
    <li>Email: <a class="email"
href="mailto:jane@doe.org">jane@doe.org</a></li>

    <li>Homepage: <a class="url"
href="http://www.yoursitehere.com/">www.yoursiteh
ere.com</a></li>
    <li>Phone: <span class="tel">+44 12345
67890</span></li>

  </ul>
  <ol class="vcalendar">
    <li class="education vevent"><a class="url
summary" href="http://example/">Example</a>
        (<abbr class="dtstart" title="2007-02-
11">2007</abbr> - <abbr class="dtend"
title="2009-03-22">2009</abbr>)</li>

  </ol>
  <ol class="vcalendar">
    <li class="experience vevent"><span
class="summary">CEO</span>, <span
class="location">Microland</span>,
        <abbr class="dtstart" title="2006-09-
01">May 2006</abbr> - <abbr title="2009-05-
22">present</abbr></li>

  </ol>
  <ul class="vcard">
    <li><a href="/jdoe/index.php" class="include"
title="Jane Doe"></a></li>

    <li class="org">MicroLand</li>
    <li class="title">CEO</li>
  </ul>
  <p>I have skills in
    <a class="skill" rel="tag"
href="http://en.wikipedia.org/wiki/HTML">HTML</a>
and
    <a class="skill" rel="tag"
href="http://en.wikipedia.org/wiki/CSS">CSS</a>.
  </p>

</div>
```

**Racine** : hResume.
Valeur d'attribut obligatoire :

- contact (hCard, adr).

Valeurs d'attributs optionnels :

- summary ;
- education (hCard, vEvent) ;
- experience (hCard, vEvent) ;
- affiliation (hCard) ;
- skills ;
- publications.

### 3.13. hReview

Vous pouvez aussi utiliser le générateur hReview plutôt que d'écrire le code à la main : Lien 66.

```
<div class="hreview">
  <p><span class="rating">5</span> out of 5
stars</p>
  <h4 class="summary">Noodle Hut</h4>

  <span class="reviewer vcard">Reviewer:
    <span class="fn">John Doe</span> - <abbr
class="dtreviewed" title="20070418T2300-
0700">April 18, 2007</abbr>

  </span>
  <p class="description item vcard">
    <span class="fn org">Noodles Hut</span> is
one of the best little places out there!
  </p>
  <ul>

    <li>Visit date: April 2007</li>
    <li>Food eaten: Instant noodles</li>
  </ul>
</div>
```

**Racine** : hReview.
Valeur d'attribut obligatoire :

- item (type (product, business, event, person, place, website, url), hCard / hCalendar).

Valeurs d'attributs optionnels :

- reviewer (hCard) ;
- version ;
- summary ;
- dtreviewed ;
- rating ;
- description ;
- tags ;
- permalink ;
- license.

### 3.14. Rel

```
<a rel="license"
href="http://creativecommons.org/licenses/by/2.0/
">Some rights reserved.</a>
<a rel="nofollow" href="http://www.w3.org/">World
Wide Web consortium</a>
```

Valeurs d'attributs :

- license ;
- nofollow ;
- tag ;
- directory ;

- enclosure ;
- home ;
- payment.

### 3.15. Robot Exclusion Profile

```
<head profile="http://example.org/xmdp/robots-
profile#"">
</head>
...
<p><img src="exclusive.png" class="robots-
noarchive" alt="Exclusive image" /></p>
```

Valeurs d'attributs :
- robots-nofollow ;
- robots-follow ;
- robots-noindex ;
- robots-index ;
- robots-noanchortext ;
- robots-anchortext ;
- robots-noarchive ;
- robots-archive.

### 3.16. VoteLinks

```
<a rev="vote-for"
href="http://www.yoursitehere.com/vote.php?
id=yes" title="Vote yes!">Vote Yes!</a>
<a rev="vote-abstain"
href="http://www.yoursitehere.com/vote.php?
id=maybe" title="Vote maybe!">Vote Maybe!</a>

<a rev="vote-against"
href="http://www.yoursitehere.com/vote.php?id=no"
title="Vote no!">Vote No!</a>
```

Valeurs d'attributs :
- vote-for ;
- vote-abstain ;
- vote-against.

### 3.17. XFN

Vous pouvez aussi utiliser le générateur XFN plutôt que d'écrire le code à la main : Lien 67.

```
<a href="http://www.yoursitehere.com" rel="me">My
Site!</a>
```

Valeurs d'attributs :
- Friendship (contact, acquaintance, friend) ;
- Physical (met) ;
- Professional (co-worker, colleague) ;
- Geographical (co-resident, neighbor) ;
- Family (child, parent, sibling, spouse, kin) ;
- Romantic (muse, crush, date, sweetheart) ;
- Identity (me).

### 3.18. xFolk

```
<ul>
  <li>
    <ul class="xfolkentry">
      <li><a class="taggedlink"
href="http://www.google.com"
title="Google">Google</a></li>

      <li class="description">The home page of
the world's biggest search engine</li>
      <li class="meta">Tags:
        <a rel="tag"
href="http://del.icio.us/tag/google">google</a>

        <a rel="tag"
href="http://del.icio.us/tag/search">search</a>
      </li>
    </ul>
  </li>

</ul>
```

**Racine** : xFolkEntry.
Valeurs d'attributs obligatoires :
- description ;
- taggedlink ;
- title.

Valeur d'attribut optionnel :
- meta (tag).

### 3.19. XMDP

```
<head
profile="http://www.mysitehere.com/profilename">
```

Racine : profile.

### 3.20. XOXO

```
<ol class="xoxo">

  <li>Subject 1
    <ol>
      <li>item a</li>
      <li>item b</li>
    </ol>

  </li>
  <li>Subject 2
    <ol>
      <li>item a</li>
      <li>item b</li>

    </ol>
  </li>
</ol>
```

**Racine** : XOXO

## 4. Conclusion

Beaucoup de microformats existent déjà et la communauté cherche toujours des manières d'utiliser des balises existantes pour valoriser davantage l'information contenue dans vos pages Web.

Ils ne présentent pas seulement un intérêt pour les moteurs de recherche et les réseaux sociaux, mais également pour les utilisateurs qui naviguent sur votre site.

*Retrouvez l'article de Sébastien Poudat en ligne : Lien 68*

## VBA et développement Web

Tour d'horizon du développement Web en VBA.

### 1. Introduction

Les applications Office ne sont certes pas dédiées à une utilisation avec internet.
Nous pouvons cependant grâce à la programmation VBA créer des passerelles entre internet et Office, en local ou en réseau.

Ce tour d'horizon de développement Web sera loin d'être exhaustif, car le sujet est vaste.

Nous allons écrire beaucoup de code VBA. Un niveau intermédiaire sur ce sujet ne sera pas superflu.

Si vous êtes débutant, commencez au moins par lire cet article : Initiation au VBA Office (Lien 69)

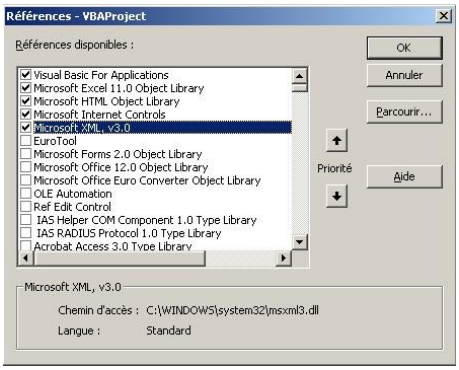
### 2. Librairies utilisées

VBA n'est pas par défaut équipé pour gérer des objets tels qu'un navigateur Internet ou un document XML.
Nous allons donc utiliser différentes librairies dont nous détaillerons l'utilisation dans les chapitres suivants.

À noter :
- à partir de la version 2010, Access dispose d'un contrôle navigateur internet standard ;
- un article à lire pour la gestion du XML à partir d'Office 2007 : Office 2007 et le XML (Lien 70).

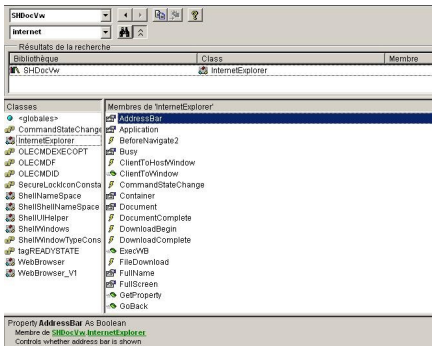
| Nom | Fichier | Description |
|---|---|---|
| SHDocVw | shdocvw.dll | Microsoft Internet Controls |
| MSXML2 | msxmlX.dll | Microsoft XML, vx.x |
| MSHTML | mshtml.tlb | Microsoft HTML Object Library |
| WinHttp | winhttp.dll | Microsoft WinHttp Services, version x.x |
| XcpControlLib | npctrl.dll | AgControl x.x Type Library (Silverlight) |

Ces librairies peuvent être ajoutées aux références du projet VBA.
Pour ajouter une référence :
- ouvrez l'éditeur VBA (ALT+F11) ;
- choisissez dans le menu : **Outils** => **Références ...** ;
- recherchez la librairie dans la liste puis cochez-la ;
- validez avec le bouton **OK**.




N'ajoutez bien attendu que les librairies nécessaires.

Une fois ajoutées aux références, les membres (propriétés, méthodes...) de ces librairies sont consultables dans l'Explorateur d'objets.
Affichez cet explorateur par le menu **Affichage** => **Explorateur d'objets** ou par le raccourci **F2**.



Sans ce référencement, les objets sont utilisables mais ils ne sont pas proposés par l'éditeur VBA.
Parfois pratique pour développer sans connaître la version de la librairie qui sera installée, ce mécanisme est connu sous le nom de lien tardif (late binding).
Les objets sont alors déclarés de type **Object** et aucune instruction n'est vérifiée à la compilation.

En plus de ces librairies, nous utiliserons aussi des API (Application Programming Interface).
Cf. les chapitres API WinInet, API URL Monikers et API Winsock.

#### 2.1. Microsoft Internet Controls

Cette librairie contient les objets :
- **Application Internet Explorer** : c'est l'application qui s'ouvre quand on exécute Internet Explorer ;

- **Navigateur Microsoft (WebBrowser)** : ce contrôle ActiveX permet d'intégrer un navigateur dans un formulaire.

Retrouvez les références (propriétés, méthodes...) de ces contrôles sur MSDN (en anglais) : Reference for Visual Basic Developers (Lien 71).

Cette librairie est présente sur le PC si Microsoft Internet Explorer 4.0 (ou plus récent) y est installé.

### 2.2. Microsoft XML, vX.X

Cette librairie est spécialisée dans le traitement des fichiers XML.
On peut charger un fichier XML et parcourir son arborescence d'objets.
Il est également possible d'y ajouter des éléments et de sauvegarder le fichier.

Une classe très utile (XMLHTTP) de cette librairie permet également d'envoyer des requêtes (GET/POST) pour télécharger une page ou un fichier par exemple.
Cette même classe donne la possibilité d'envoyer des données à un script PHP pour remplir un formulaire et recevoir un résultat en retour.

Pour connaître la version de la librairie XML disponible en fonction de l'OS et des applications installées, rendez-vous sur cette page : Liste des versions de Microsoft XML Parser (MSXML) (Lien 72).

### 2.3. Microsoft HTML Object Library

Cette librairie référence les objets du modèle HTML (document, formulaire, tableau...).
Elle permet de parcourir le contenu d'une page HTML.

Vous trouverez les références (propriétés, méthodes...) (en anglais) de cette librairie sur MSDN : MSHTML Reference (Lien 73).

Cette librairie est présente sur le PC si Microsoft Internet Explorer 4.0 (ou plus récent) y est installé.

### 2.4. Microsoft WinHTTP Services, vX.X

Cette librairie regroupe les fonctions HTTP des API WinInet.

Vous trouverez les références (propriétés, méthodes...) (en anglais) de cette librairie sur MSDN : WinHttpRequest Object (Lien 74).

Pour connaître la version de la librairie XML disponible en fonction de l'OS et des applications installées, rendez-vous sur cette page : WinHTTP Versions (Lien 75).

### 2.5. AgControl X.X Type Library (Silverlight)

Cette librairie référence les objets nécessaires à l'utilisation du plugin Silverlight (Lien 76).

## 3. Contrôles internet

Pour dialoguer avec internet, il paraît naturel de vouloir piloter un navigateur tel qu'**Internet Explorer**.

Dans cet article, nous n'aborderons que ce navigateur.

### 3.1. Créer un navigateur internet externe à l'application

Le navigateur se présente en VBA comme un objet de type **InternetExplorer**.
Cet objet représente un navigateur internet tel qu'il se présente lorsqu'on exécute Internet Explorer.

Pour disposer dans VBA des instructions nécessaires, il faut d'abord ajouter la librairie **Microsoft Internet Controls**.

Ensuite créez dans l'éditeur VBA un nouveau module de code : **Insertion** => **Module**.

Avant de créer le module, je vous conseille de modifier l'option de déclaration des variables :
- dans le menu : **Outils** => **Options...** ;
- cochez la case **Déclaration des variables obligatoire**.

Cela a pour effet d'ajouter en en-tête de chaque nouveau module l'instruction Option Explicit.
Cette instruction impose la déclaration de toutes les variables utilisées dans le code.
En précisant le type exact de toutes nos variables, on évite des conversions inutiles de données.
De plus, sans cette option, il est fréquent de faire des erreurs d'étourderie difficiles à déceler (une faute de frappe dans un nom de variable par exemple).
Il est également possible d'écrire cette instruction Option Explicit manuellement.

Nous allons écrire une petite procédure :

```
Public Sub CreerNavigateur()

End Sub
```

Dans cette procédure, commençons par déclarer notre objet navigateur :

```
Dim oNav As SHDocVw.InternetExplorer
```

Il n'y parait pas, mais il y a déjà beaucoup à dire sur cette ligne de code.

D'abord vous noterez que notre variable **oNav** qui contient un objet est préfixée par la lettre o.
Ce n'est pas grand-chose mais il est souvent utile de voir en un coup d'œil sur un code assez complexe de quel type est une variable.
Je vous renvoie à ces deux articles pour plus de détails sur les conventions de nommage des variables :
Descriptif des conventions typographiques du code Visual Basic : Lien 77 ;
Convention générale de codage pour la programmation : Lien 78.

Ensuite on sait que l'objet que l'on souhaite créer est défini dans la librairie **SHDocVw** (Microsoft Internet Controls).
Il suffit alors de taper un point après ce nom de librairie pour avoir la liste de toutes les classes de cette librairie.
On pourrait ne pas préfixer la classe **InternetExplorer** par

sa librairie **SHDocVw**.
Auquel cas VBA rechercherait dans toutes les références actives du projet une classe de nom **InternetExplorer**, au risque d'utiliser une classe d'une autre librairie.
À ma connaissance, **InternetExplorer** n'est pas défini dans une autre librairie qu'on utiliserait fréquemment, et donc le nom de librairie n'est pas ici indispensable.

Si vous n'êtes pas familier avec les variables objets, sachez qu'on a seulement déclaré notre variable **oNav** pour une utilisation future.
Elle n'est pour l'instant liée à aucun objet, le test oNav is Nothing renvoie Vrai.

Pour créer l'objet, il suffit d'utiliser le mot-clé **New**.

```
Set oNav = New SHDocVw.InternetExplorer
```

Il faut que le type qui suit **New** soit le même que celui utilisé pour déclarer la variable.

**oNav** pointe désormais vers un navigateur Internet Explorer.
À noter : ce navigateur est par défaut invisible !
Si vous exécutez le code tel quel, vous allez créer un navigateur qui restera ouvert. On ne pourra le fermer que dans le gestionnaire des tâches.

Garder le navigateur invisible peut être utile pour effectuer un traitement en arrière-plan.
Utilisez alors la méthode **Quit** de l'objet pour le fermer proprement.

Pour voir le navigateur, définissez sa propriété **Visible**.

```
oNav.Visible = True
```

Exécutez maintenant la procédure (**F5** avec le curseur à l'intérieur de la procédure).
Une application **Internet Explorer** s'ouvre.

Par défaut, aucune page n'est ouverte.
Demandons la navigation vers une page : Google par exemple.
Il suffit d'utiliser la méthode **Navigate** de notre objet.

```
oNav.Navigate "http://google.fr"
```

## 3.2. Intégrer un navigateur internet à l'application

Le navigateur tel que présenté dans le chapitre précédent s'ouvre dans une fenêtre distincte de l'application Office.
Il est également envisageable d'intégrer un navigateur dans un formulaire.
On parle alors de contrôle **ActiveX** WebBrowser (ou Navigateur Web).
Pour disposer dans VBA du contrôle et des instructions nécessaires, il faut d'abord ajouter la librairie **Microsoft Internet Controls**.

La méthode de création du contrôle est légèrement différente selon l'application et le type de formulaire.

### 3.2.1. Intégrer un navigateur internet à un formulaire Access

Avec un formulaire Access, le WebBrowser s'intègre dans un cadre d'objet qui contient l'ActiveX.

Insérez un nouveau contrôle ActiveX :
- avant Access 2007 : menu **Insertion** => **Contrôle ActiveX** ;
- à partir d'Access 2007 : onglet **Création** => groupe **Contrôles** => **Contrôle ActiveX**.

Recherchez le contrôle **Navigateur Web Microsoft** (ou **Microsoft Web Browser**) et placez-le sur le formulaire.
Par défaut le premier contrôle est nommé **WebBrowser0** ; nous le renommons en **ctlNav** par exemple.
Si vous ne l'avez pas déjà fait, la librairie **Microsoft Internet Controls** est automatiquement sélectionnée dans les références.

Une fois le contrôle créé, on dispose d'un objet de type **WebBrowser** (ou Navigateur Web) qui se pilote de manière similaire à un objet **InternetExplorer**.

Nous allons écrire le code de navigation vers Google à l'ouverture du formulaire.

Afficher les propriétés du formulaire :
- avant Access 2007 : menu **Affichage** => **Propriétés...** ;
- à partir d'Access 2007 : onglet **Création** => groupe **Outils** => **Feuille des propriétés**.

Sélectionnez l'événement **Sur Ouverture**, choisissez **[Procédure événementielle]** dans la liste déroulante et cliquez sur les trois petits points à droite.

La procédure **Form_Open** est générée ; écrivons notre code à l'intérieur.

```
Private Sub Form_Open(Cancel As Integer)
   Me.ctlNav.Object.Navigate "http://google.fr"
End Sub
```

Le contrôle ctlNav est un cadre d'objet qui contient l'activeX WebBrowser

Pour atteindre le contrôle WebBrowser, il faut passer par la propriété **Object** du cadre d'objet.
**Object** n'étant pas d'un type particulier, il n'y a pas d'autocomplétion.
Pour retrouver cette fonctionnalité bien pratique, il faut déclarer un objet de type WebBrowser et le lier à notre contrôle.

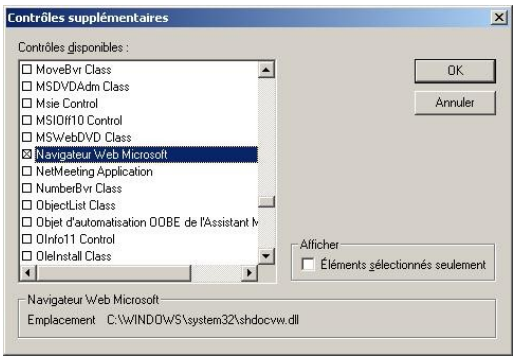
```
Dim oNav As SHDocVw.WebBrowser
Set oNav = Me.ctlNav.Object
oNav.Navigate "http://google.fr"
```

Affichez le formulaire pour voir la page affichée dans le WebBrowser.

Seul l'intérieur de la fenêtre est affiché. Les barres d'outils et de statut ne sont pas affichées.

### 3.2.2. Intégrer un navigateur internet à un formulaire utilisateur (UserForm)

Pour Excel, Word ou encore PowerPoint, les formulaires disponibles sont appelés des **UserForms**.
Créez un nouveau formulaire à partir de l'éditeur VBA : menu **Insertion** => **UserForm**.
Dans le menu **Outils** => **Contrôles supplémentaires...**, recherchez **Navigateur Web Microsoft** et cochez-le.




Le contrôle WebBrowser est alors disponible dans la boîte à outils (icône globe).
Si la boîte à outils n'est pas affichée, réaffichez-la par le menu : **Affichage** => **Boîte à Outils**.


Cliquez sur ce bouton et sélectionnez la zone du formulaire sur laquelle déposer le contrôle.

Par défaut, le premier contrôle est nommé **WebBrowser1**.
Vous pouvez le renommer en **ctlNav** par exemple dans la Fenêtre Propriétés (F4).

Une fois le contrôle créé, on dispose d'un objet de type **WebBrowser** qui se pilote de manière similaire à un objet **InternetExplorer**.

Nous allons écrire le code de navigation vers Google à l'ouverture du formulaire.
Passez en affichage de code (menu **Affichage** => **Code**) et sélectionnez dans les listes déroulantes en haut de module :
**UserForm** à gauche puis **Initialize** à droite.
La procédure **UserForm_Initialize** est générée ; écrivons notre code à l'intérieur.

```
Me.ctlNav.Navigate "http://google.fr"
```

Affichez le formulaire (F5) pour voir la page affichée dans le WebBrowser.

Seul l'intérieur de la fenêtre est affiché. Les barres d'outils et de statut ne sont pas affichées.

### 3.3. Piloter un navigateur internet

Les contrôles InternetExplorer et WebBrowser exposent des méthodes utiles à la navigation :
- **Navigate** pour naviguer vers une adresse ;
- **GoBack** pour naviguer vers la page précédente ;
- **GoForward** pour naviguer vers la page suivante ;
- **GoHome** pour naviguer vers la page d'accueil ;
- **Refresh** pour rafraîchir la page en cours.

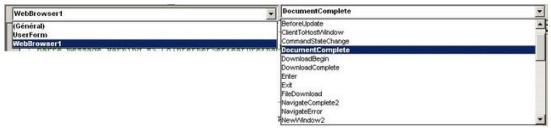
Ces méthodes, associées à des boutons sur le formulaire, permettent une navigation minimale dans un WebBrowser sans avoir besoin du menu complet.
Elles sont également utilisables sur une application InternetExplorer.

### 3.4. Événements d'un navigateur internet

Les navigateurs exposent différents événements.
Vous trouverez la documentation de ces événements sur MSDN (en anglais) : Reference for Visual Basic Developers (Lien 79).

#### 3.4.1. Événements d'un contrôle WebBrowser

Pour un contrôle **WebBrowser**, ils sont facilement accessibles dans le code du formulaire.
Sélectionnez d'abord le contrôle dans la liste déroulante en haut à gauche de la page de code (**ctlNav** par exemple).
Puis choisissez l'événement dans la liste déroulante de droite (**DocumentComplete** par exemple).



La procédure est automatiquement générée :

```
Private Sub ctlNav_DocumentComplete(ByVal pDisp
As Object, URL As Variant)
End Sub
```

**URL** contient l'adresse de la page, **pDisp** est un objet qui représente le navigateur.
Ajoutons une boîte de message qui s'affichera à chaque chargement de page.

```
Private Sub ctlNav_DocumentComplete(ByVal pDisp
As Object, URL As Variant)
   MsgBox "Document chargé" & vbCrLf & URL
End Sub
```

#### 3.4.2. Événements d'un contrôle InternetExplorer

Pour un "contrôle" **InternetExplorer**, on peut accéder aux événements en déclarant ce contrôle avec l'instruction **WithEvents**.
Il est alors nécessaire que le code soit écrit dans un module de classe (la page code d'un formulaire est aussi un module de classe).

Voici un exemple avec l'événement **DocumentComplete**.
Ce code est écrit dans le module d'un formulaire contenant un bouton **btnNavigue**.
Sur clic sur le bouton, on crée un navigateur est on charge la page de recherche Google.

```
Private WithEvents oNav As
SHDocVw.InternetExplorer

Private Sub btnNavigue_Click()
Set oNav = New SHDocVw.InternetExplorer
oNav.Visible = True
oNav.navigate "http://google.fr"
End Sub

Private Sub oNav_DocumentComplete(ByVal pDisp As
Object, URL As Variant)
   MsgBox "Document chargé" & vbCrLf & URL
End Sub
```

Le fait de déclarer le navigateur avec **WithEvents** active les événements et permet de les choisir dans les listes déroulantes en haut de la page de code.

## 4. Librairie HTML

Jusqu'ici nous avons navigué sur une page sans regarder son contenu.
Une page internet est codée en langage HTML.
Si vous ne connaissez pas ce langage, vous pouvez consulter ce tutoriel : Les bases du HTML (Lien 80).

Ce langage étant structuré, nous allons lire sa structure à l'aide de la librairie **Microsoft HTML Object Library**.
Commençons donc par référencer cette librairie dans le menu : **Outils** => **Références ...**.

### 4.1. Parcourir un document HTML

Afin de parcourir un document HTML en provenance d'internet par exemple, il faut d'abord le charger.
Chargeons la page de recherche Google dans un navigateur.
Utilisons pour cela le code vu dans le chapitre précédent :

```
Public Sub CreerNavigateur()
Dim oNav As SHDocVw.InternetExplorer
Set oNav = New SHDocVw.InternetExplorer
oNav.Visible = True
oNav.navigate "http://google.fr"
End Sub
```

Cette fonction ouvre Internet Explorer et navigue vers la page de recherche de Google.
Il est également possible d'ouvrir la page dans un contrôle WebBrowser, la suite du code serait identique.

Vous pouvez visualiser la source en cliquant sur la page avec le bouton droit et en choisissant "Afficher la source".
Le code affiché est le code HTML ; il est difficilement exploitable en l'état par code VBA.

En fonction de votre version d'Internet Explorer, vous avez peut-être accès aux outils de développements (dans le menu Outils ou F12).
Cet outil s'avère très pratique pour analyser une page HTML.

Les contrôles navigateurs ont une propriété **Document** de type Object.
Cette propriété est une passerelle entre la librairie **Microsoft Internet Controls** et la librairie **Microsoft HTML Object Library**.

C'est en fait un objet de type **HTMLDocument** référencé dans la librairie HTML.
Pour profiter de l'autocomplétion de VBA, nous allons déclarer un objet de ce type afin d'y affecter la valeur de cette propriété.

```
Public Sub CreerNavigateur()
Dim oNav As SHDocVw.InternetExplorer
Set oNav = New SHDocVw.InternetExplorer
oNav.Visible = True
oNav.navigate "http://google.fr"
Set oDoc = oNav.document
End Sub
```

Si on exécute le code, on obtient une erreur :

```
Erreur d'exécution '-2147467259(80004005)':

Erreur Automation
Erreur non spécifiée
```

Cette erreur est due au fait que l'on essaye de lire la propriété **Document** avant que celui ne soit chargé.
Il faut donc ajouter une temporisation afin d'attendre le chargement de la page.

Pour vérifier l'état du navigateur, il y a deux propriétés :
- **ReadyState** : indique l'état de chargement de la page (en cours, chargée...)
- **Busy** : indique si le navigateur est occupé.

Il est possible que le chargement soit terminé, mais que le navigateur soit toujours occupé à faire diverses tâches.
Nous allons utiliser ces deux propriétés pour nous assurer que tout est bien chargé et que les objets HTML sont tous présents dans l'objet Document.
Une simple boucle suffirait :

```
While oNav.readyState <> READYSTATE_COMPLETE Or
oNav.Busy = True
    DoEvents
Wend
```

Ce code attend que le chargement de la page soit terminé et que le navigateur ne soit plus occupé.
**DoEvents** peut être nécessaire pour que certains éléments se chargent.
Sans **DoEvents**, il est possible qu'on reste bloqué dans une boucle sans fin.
Mais comme nous aurons probablement besoin de faire cette même temporisation plusieurs fois, nous allons créer une petite fonction.
Nous en profiterons pour ajouter un timeout qui évitera une boucle sans fin en cas de blocage du navigateur.

```
' Attend que la page internet soit chargée
' pTimeOut est un time out en secondes (WaitIE
vaut True si Timeout)
Public Function WaitIE(oIE As InternetExplorer,
Optional pTimeOut As Long = 0) As Boolean
Dim lTimer As Double
lTimer = Timer
Do
    DoEvents
   If oIE.readyState = READYSTATE_COMPLETE And
Not oIE.Busy Then Exit Do
```

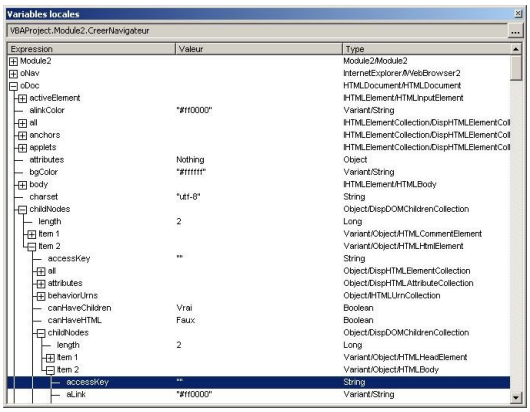
```
   If pTimeOut > 0 And Timer - lTimer > pTimeOut
Then
       WaitIE = True
       Exit Do
   End If
Loop
End Function
```

Cette fonction peut être placée dans n'importe quel module VBA de l'application.
On lui donne en paramètre l'objet Navigateur que l'on veut attendre, et un timeout en secondes.
Si le navigateur n'a pas rendu la main à l'issue du timeout, la fonction renvoie True.

```
Public Sub CreerNavigateur()
Dim oNav As SHDocVw.InternetExplorer
Dim oDoc As MSHTML.HTMLDocument
Set oNav = New SHDocVw.InternetExplorer
oNav.Visible = True
oNav.navigate "http://google.fr"
' Attente avec timeout de 10 s
If WaitIE(oNav, 10) Then
   ' 10 s écoulées et page non chargée
   MsgBox "Time out!"
Else
   ' Page chargée, on continue
   Set oDoc = oNav.document
End If
End Sub
```

Cette fonction WaitIE pourra être appelée à chaque navigation.
Si on a précisé un timeout et qu'elle renvoie True, on ne continue pas le traitement.

Sinon, on charge le document dans l'objet **oDoc**.
Pour visualiser cet objet, placez un point d'arrêt (F9 sur la ligne End If), ou ajoutez une instruction Stop.
Exécutez ensuite la fonction, l'exécution s'arrête.
Affichez la fenêtre **Variables Locales** : **Affichage** => **Fenêtre Variables Locales**.




Vous pouvez naviguer dans l'arborescence de l'objet **oDoc**.
Chaque élément possède une collection **ChildNodes** qui permet de visualiser hiérarchiquement tous les objets HTML du document.
Dans la collection **All**, tous les éléments de l'arborescence sont mis à plat.
Il existe d'autres collections regroupant des objets de même type, par exemple :
forms (formulaires), links (hyperliens), images (images)...

Le nombre d'éléments dans une collection est donné par la propriété **Length** :
- oDoc.Images.Length => 4
Le premier élément est l'élément d'indice 0 :
- oDoc.Images(0).src => http://www.google.fr/images/close_sm.gif

Pour chercher un élément par son Identifiant, on utilisera la fonction **getElementById**.
Pour chercher un élément par son Nom, on utilisera la fonction **getElementsByName**.

*Retrouvez la suite de l'article d'Arkham46 en ligne : Lien 81*

## L'opérateur SQL IN : Comment utiliser une liste d'éléments dans une requête

L'opérateur IN, à la différence de LIKE, n'est disponible que pour SQL. Au vu de l'interaction de ce langage avec ACCESS, il peut vous rendre de précieux services et vous éviter des expressions à rallonge.

### 1. Introduction

L'opérateur logique **IN** est un opérateur d'égalité strictement réservé à SQL. Il permet de déterminer si la valeur d'une expression est égale à l'une des valeurs comprises dans une liste donnée.

L'opérateur **IN** ne doit être confondu ni avec la clause de SQL qui détermine une source de données ni avec le mot clef VBA de la syntaxe For.

### 2. La syntaxe

La syntaxe est la suivante :

```
WHERE NomChamp IN (valeur1, valeur2, ... , valeurX )
```

La liste d'éléments à comparer est placée entre parenthèses.

On peut également trouver sa forme négative :

```
WHERE NomChamp NOT IN (valeur1, valeur2, ... , valeurX )
```

Il faut savoir que dans sa forme négative, il ne bénéficie pas de l'optimisation **Rushmore**. Pour en savoir plus sur la technologie **Rushmore** consultez le tutoriel sur l'optimisation des bases de données Microsoft Access : Lien 82.

On peut également utiliser une requête comme liste d'éléments à comparer, à condition que cette dernière ne renvoie qu'un champ.

```
SELECT * FROM TableA WHERE NomChamp IN (SELECT ChampId FROM TableB)
```

Cette forme de relation est intéressante par la souplesse de sa rédaction.

### 1.1. IN vs. =

Les exemples suivants démontrent l'avantage de l'opérateur **IN**.

Avec l'opérateur d'égalité la requête aura cette forme :

```
SELECT * FROM TableA WHERE champId = 1 or champId = 3 or champId = 10;
```

La même requête simplifiée grâce à l'opérateur **IN** :

```
SELECT * FROM TableA WHERE champId IN (1,3,10);
```

### 3. En pratique

Nous allons voir comment l'opérateur s'intègre parfaitement au fonctionnement d'une application pour créer des filtres à moindre coût.

### 3.1. Zone liste, IN et état

La zone de liste est un contrôle sympathique, encore faut-il pouvoir tirer parti de sa sélection multiple.

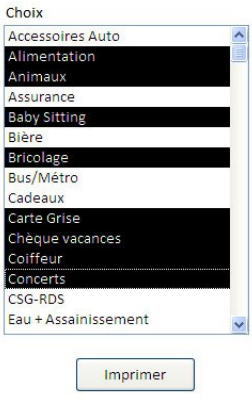
Pour la mise en pratique vous avez besoin d'une table contenant au moins un identifiant unique de type numérique et un champ de type texte.

Commencez par créer un formulaire composé d'une zone de liste et d'un bouton de commande.

La zone de liste doit être réglée de la manière suivante :

| | |
|---|---|
| Nom | Zliste |
| Nbre Colonnes | 2 |
| Largeurs colonnes | 0 ;5 |
| Contenu | SELECT TCategorie.IdCategorie, TCategorie.Nom FROM TCategorie ORDER BY TCategorie.Nom; |
| Sélection Multiple | Simple (ou Étendue) |

Voici le résultat.




Créez ensuite un état nommé **Test** contenant les champs de cette même table. Peu importe la mise en page, c'est pour l'exemple.

Grâce à l'opérateur **IN** nous allons imprimer les éléments choisis dans la liste.

Dans l'événement **Sur clic** du bouton de commande insérez le code suivant :

```
Private Sub Commande2_Click()
Dim itm As Variant          ' l'item choisi
Dim lstval As String        ' la liste des id sélectionnés
For Each itm In Me.Zliste.ItemsSelected
' parcourt les items
    lstval = lstval & Me.Zliste.ItemData(itm) & ","     ' insère chaque id dans la variable
Next
lstval = Left(lstval, Len(lstval) - 1)
' supprime la dernière virgule
DoCmd.OpenReport "Test", acViewPreview, , "IdCategorie IN (" & lstval & ")"  ' imprime la sélection
End Sub
```

La méthode est simple, cependant elle ne gère pas l'erreur levée lorsqu'il n'y a pas de choix.

Sélectionnez quelques éléments et cliquez sur le bouton.

| IdCategorie | Nom |
|---|---|
| 1 | Carte Grise |
| 6 | Alimentation |
| 19 | Chèque vacances |
| 36 | Concerts |
| 37 | Baby Sitting |
| 38 | Bricolage |
| 50 | Animaux |
| 52 | Coiffeur |

### 3.1.1. Variante avec du texte

La syntaxe est pratiquement identique à une différence près. Comme nous utilisons des éléments texte il faut utiliser le séparateur texte dans la liste de l'opérateur **IN**.

Changez la propriété de la liste :

| | |
|---|---|
| Colonne Liée | 2 |

Modifiez le code du bouton :

```
lstval = lstval & """ & Me.Zliste.ItemData(itm) & ""","    ' insère chaque id dans la variable
```

Modifiez l'appel de l'état pour intégrer le nouveau nom de champ.

```
DoCmd.OpenReport "Test", acViewPreview, , "Nom IN (" & lstval & ")"  ' imprime la sélection
```

La syntaxe SQL sera donc :

```
WHERE NOM IN ("coiffeur","concerts" )
```

### 3.1.2. Variante avec des dates

Construisez un nouvel état avec cette source contenant une date et un champ texte. Indiquez la même source pour la zone de liste.

La ligne de construction de la liste change. Il faut mettre le symbole dièse (#) qui est le séparateur de dates et le format de date US.

```
lstval = lstval & "#" & Format(Me.ZListe.ItemData(itm), "mm/dd/yyyy") & "#,"     ' insère chaque date
```

Les dates dans VBA doivent toujours être au format anglo-saxon.

L'appel du formulaire change également pour utiliser le bon nom de champ.

```
DoCmd.OpenReport "Test2", acViewPreview, , "DateOperation IN (" & lstval & ")"  ' imprime la sélection
```

La syntaxe SQL sera donc :

```
WHERE DateOperation IN (#10/02/2011#,#12/12/2010#)
```

## 4. Limitation

Hormis le problème avec la forme négative de l'opérateur **IN** (**Not IN**) les limitations ne sont pas légion. En effet la liste peut compter plusieurs centaines d'éléments.

La valeur **NULL** doit être considérée avec précaution. Comparée à un élément de la liste elle peut produire des effets inattendus.

La liste est un ensemble de 1 à x éléments. On peut donc utiliser ce code quel que soit le nombre de sélections.

## 5. Conclusion

Utiliser l'opérateur **IN** simplifie grandement la construction des requêtes SQL. Il évite de longs et complexes algorithmes de construction de clause **WHERE** ou **HAVING** et facilite l'écriture des requêtes de vérifications de clefs externes.

*Retrouvez l'article de Fabrice Constans en ligne : Lien 83*

# Windows Phone 7 Mango : découvrez les nouvelles tâches

Le but de cet article est de présenter les nouvelles tâches apportées par Windows Phone 7 Mango.
Chaque tâche sera vue en détail et accompagnée d'un exemple de code et de plusieurs captures d'écran.

## 1. Introduction

Lorsque l'on développe une application mobile, il est bien souvent nécessaire, et même recommandé, d'utiliser les fonctionnalités de base de l'OS utilisé. Cela permet d'adapter l'application à la plateforme ciblée, mais également de gagner du temps lors du développement puisque, la plupart du temps, les fonctionnalités en question sont accessibles depuis des API fournies par les éditeurs. Avec Windows Phone 7, Microsoft ne déroge pas à la règle et fournit un ensemble d'interfaces de programmation permettant d'accéder aux fonctionnalités les plus utilisées de son système d'exploitation mobile.

Avec Mango, ou Windows Phone 7.5, le géant du logiciel apporte un bon nombre de nouveautés. De ce fait, les outils de développement évoluent également et il est dorénavant possible de réaliser de nombreuses choses qui étaient jusqu'alors impossibles. Pour faciliter l'interaction avec les fonctionnalités de base de Windows Phone 7, Microsoft met à la disposition des développeurs le namespace Microsoft.Phone.Tasks. Ce dernier regroupe la totalité des tâches (tasks) disponibles, c'est-à-dire l'ensemble des classes utilisables pour interagir avec les applications natives de Windows Phone 7.

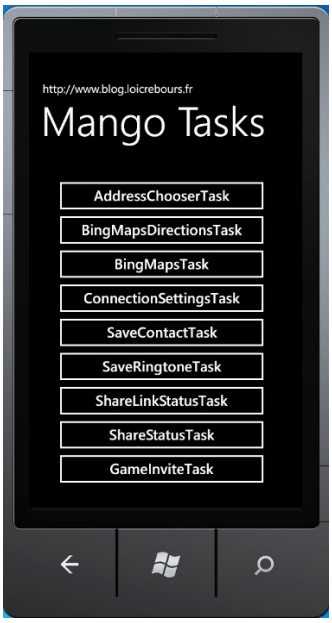
Parmi ces tâches, la première version des outils de développement,(celle visant Windows Phone 7.0),proposait notamment PhoneCallTask, SmsComposeTasks ou encore WebBrowserTask :, des classes permettant respectivement de passer un appel, d'envoyer un SMS et de lancer le navigateur. Mango apporte avec elle neuf nouvelles tâches. De l'itinéraire Bing Maps au partage d'un statut sur les réseaux sociaux, ces nouvelles tâches viennent compléter les premières pour le plus grand bonheur des développeurs.

## 2. Les nouvelles tâches

Sans plus attendre, voici un tableau récapitulant l'ensemble des nouvelles tâches apportées par Mango. Certaines s'avéreront rapidement très utiles, voire indispensables, tandis que d'autres seront plus spécifiques.

| Nom | Description |
|---|---|
| AddressChooserTask | Permet de sélectionner un contact afin de récupérer son adresse |
| BingMapsDirectionsTask | Trace un itinéraire sur Bing Maps |
| BingMapsTask | Lance une recherche sur Bing Maps |
| ConnectionSettingsTask | Active ou désactive une connexion |
| GameInviteTask | Permet d'inviter un contact pour jouer à plusieurs |
| SaveContactTask | Sauvegarde un contact dans le carnet de contacts |
| SaveRingToneTask | Sauvegarde une sonnerie dans la liste des sonneries personnalisées |
| ShareStatusTask | Partage un statut sur les réseaux sociaux liés au compte de l'utilisateur |
| ShareLinkTask | Partage un lien sur les réseaux sociaux liés au compte de l'utilisateur |

Pour chaque tâche, un exemple de code sera présenté et accompagné d'une ou plusieurs images dans la suite de l'article. Le code complet est regroupé dans une application dont les sources sont disponibles en fin d'article.




Avant de passer à la présentation individuelle de chacune des tâches, voici quelques informations qu'il est bon de connaître avant de poursuivre :

- les exemples ci-dessous nécessitent l'ajout préalable (using / import) du namespace Microsoft.Phone.Tasks ;
- chaque tâche est exécutée grâce à la méthode Show() ;
- les tâches effectuant des appels asynchrones possèdent un évènement Completed auquel il est nécessaire de souscrire ;
- l'énumération TaskEventArgs.TaskResult permet de tester le résultat d'une requête.

### 2.1. AddressChooserTask

AddressChooserTask, comme son nom le laisse entendre, permet de récupérer l'adresse d'un contact. Le comportement de cette tâche est le même que PhoneNumberChooserTask et EmailAddressChooserTask. En appelant cette tâche, Windows Phone 7 va lancer l'application Contacts et vous permettre d'en sélectionner un. Ceci fait, l'adresse du contact sélectionné vous sera renvoyée.

La classe AddressChooserTask :

**AddressChooserTask**

| | |
|---|---|
| Completed | Se déclenche lorsqu'un contact est sélectionné |
| Show | Exécute la tâche |

Code :

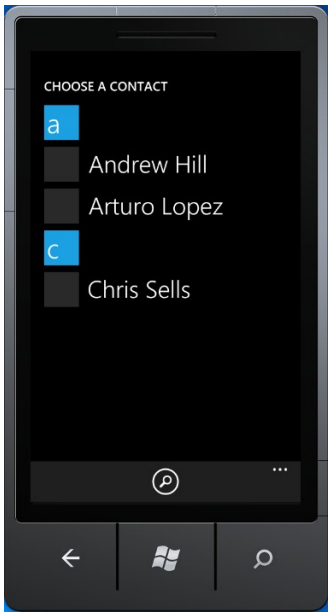
```
private void
AddressChooserTaskButton_Click(object sender,
RoutedEventArgs e)
{
   AddressChooserTask addressChooser = new
AddressChooserTask();

   addressChooser.Completed += new
EventHandler<AddressResult>(addressChooser_Comple
ted);
   addressChooser.Show();
}

void addressChooser_Completed(object sender,
AddressResult e)
{
   if (e.TaskResult == TaskResult.OK)
       MessageBox.Show(String.Format("{0} vit à
{1}", e.DisplayName, e.Address));
   else
       MessageBox.Show("Erreur lors de la
récupération de l'adresse");
}
```

Dans un premier temps, comme précisé ci-dessus, l'application Contacts est lancée. L'utilisateur voit alors apparaître l'ensemble de ses contacts et peut en sélectionner un.


Une fois le contact sélectionné, l'événement Completed est levé et l'adresse du dit contact peut être récupérée et affichée.




Un exemple d'utilisation :

Cette tâche peut trouver son utilité dans une application géolocalisant vos contacts. Une fois l'adresse récupérée, il est alors tout à fait possible d'utiliser le service Geocoding de Bing Maps afin de récupérer la latitude et longitude correspondant à cette adresse et positionner ce point sur un module Bing Maps.

## 2.2. BingMapsDirectionsTask

BingMapsDirectionsTasks permet de lancer l'application Bing Maps de votre Windows Phone 7 afin de tracer un itinéraire. Cet itinéraire est déterminé par une position initiale et une position d'arrivée qu'il est nécessaire d'indiquer à la tâche. Ces positions sont représentées par des objets de type LabeledMapLocation. Le constructeur de cet objet prend en paramètres une chaîne de caractères représentant le lieu, ainsi qu'un objet de type System.Device.Location.GeoCoordinate indiquant la position géographique du lieu en question.

La classe BingMapsDirectionsTask :

**BingMapsDirectionsTask**

| | |
|---|---|
| Start | LabeledMapLocation représentant le point de départ |
| End | LabeledMapLocation représentant le point d'arrivé |
| Show | Exécute la tâche |

Code :

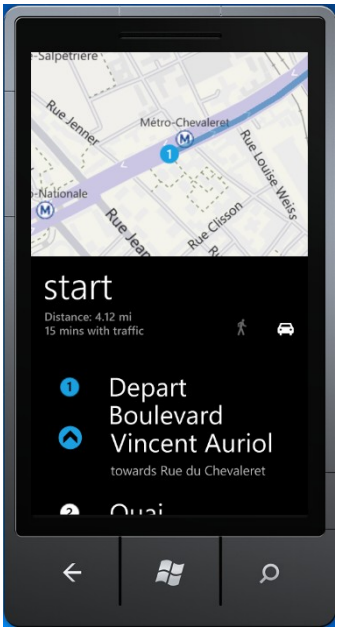
Dans l'exemple ci-dessous, j'utilise cette tâche afin de tracer l'itinéraire entre le Parc de Montsouris et le Grand Palais, à Paris.

```
private void
BingMapsDirectionsTaskButton_Click(object sender,
RoutedEventArgs e)
{
   BingMapsDirectionsTask bingMapsDirection = new
BingMapsDirectionsTask()
   {
       Start = new LabeledMapLocation("Parc de
Montsouris", new
System.Device.Location.GeoCoordinate(48.833651,
2.367039)),
```

```
        End = new LabeledMapLocation("Grand
Palais", new
System.Device.Location.GeoCoordinate(48.862597,
2.315154))
    };

    bingMapsDirection.Show();
}
```

Une fois la tâche exécutée, l'itinéraire est tracé et les différentes instructions apparaissent à l'écran.




Un exemple d'utilisation :

Cette tâche peut être particulièrement utile pour les applications géolocalisant les restaurants, magasins et autres établissements géographiquement proches de l'utilisateur. Il peut alors être envisageable d'utiliser cette tâche pour tracer rapidement un itinéraire entre l'utilisateur et l'établissement sélectionné.

### 2.3. BingMapsTask

BingMapsTasks permet de lancer l'application BingMaps afin d'effectuer une recherche quelconque. Pour cela, il est nécessaire d'indiquer les termes recherchés, le centre géographique de la recherche, représenté par un objet de type GeoCoordinate, ainsi que le niveau de zoom du module Bing Maps.

La classe BingMapsTask :

**BingMapsTask**

| | |
|---|---|
| Center | Geocoordinate représentant le point central de la recherche |
| SearchTerm | Termes recherchés |
| ZoomLevel | Niveau de zoom par défaut de Bing Maps |
| Show | Exécute la tâche |

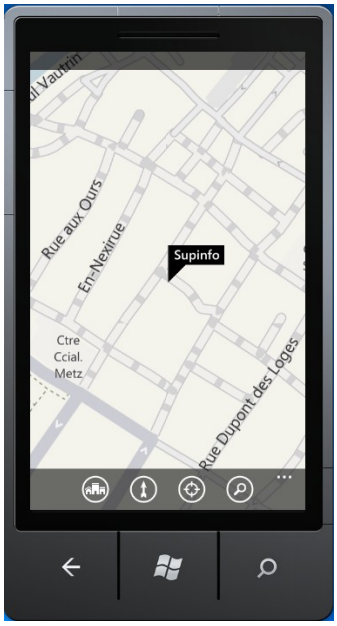
Code :

Dans l'exemple ci-dessous, je positionne le centre de ma recherche sur Metz et recherche “Supinfo Metz” avec un niveau de zoom élevé.

```
private void BingMapsTaskButton_Click(object
sender, RoutedEventArgs e)
{
    BingMapsTask bingMapsTask = new BingMapsTask()
    {
        Center = new
System.Device.Location.GeoCoordinate(49.117545,
6.174409),
        SearchTerm = "Supinfo Metz",
        ZoomLevel = 17
    };

    bingMapsTask.Show();
}
```

Une fois la recherche effectuée, Bing Maps s'ouvre et affiche l'élément recherché.




Un exemple d'utilisation :

En reprenant l'exemple d'utilisation de BingMapsDirectionsTask, cette tâche pourrait permettre à l'utilisateur de lancer manuellement une recherche sur un établissement qui n'aurait pas été trouvé par l'application.

### 2.4. ConnectionSettingsTask

ConnectionSettingsTask permet de lancer l'utilitaire qui peut activer ou désactiver (on/off) une connexion. Cette connexion est déterminée par l'énumération Microsoft.Phone.Tasks.ConnectionSettingsType et prend l'une des valeurs suivantes :

- AirplaneMode ;
- Bluetooth ;
- Cellular ;
- WiFi.

La classe ConnectionSettingsTasks :

**ConnectionSettingsTask**

| | |
|---|---|
| ConnectionSettingsType | Type de connexion à activer ou désactiver |
| Show | Exécute la tâche |

Code :

Dans l'exemple ci-dessous, j'exécute l'application permettant d'activer ou désactiver le WiFi. Pour cela, il suffit d'utiliser la valeur WiFi de l'énumération ConnectionSettingsType et d'appeler la méthode Show() de la tâche.

```
private void
ConnectionSettingsTaskButton_Click(object sender,
RoutedEventArgs e)
{
   ConnectionSettingsTask connectionSettings =
new ConnectionSettingsTask()
   {
       ConnectionSettingsType =
ConnectionSettingsType.WiFi
   };

   connectionSettings.Show();
}
```

L'application s'exécute aussitôt.


Un exemple d'utilisation :

Beaucoup d'applications nécessitent une connexion pour fonctionner convenablement. Cette tâche peut alors être utilisée pour permettre à l'utilisateur d'activer rapidement ses connexions si cela n'est pas déjà fait.

### 2.5. GameInviteTask

GameInviteTask lance l'écran d'invitation permettant à un utilisateur d'inviter d'autres joueurs à une session multijoueur. Pour exécuter cette tâche, seul un identifiant de session est nécessaire.

La classe GameInviteTask :

**GameInviteTask**

| | |
|---|---|
| Completed | Se déclenche après l'envoi de l'invitation |
| SessionId | Identifiant de la session |
| Show | Exécute la tâche |

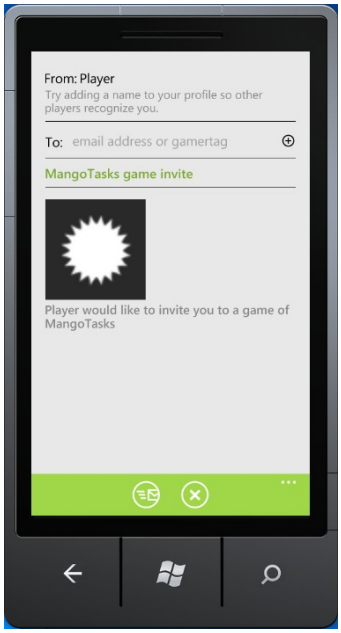
Code :

```
private void GameInviteTaskButton_Click(object
sender, RoutedEventArgs e)
{
   GameInviteTask gameInvite = new
GameInviteTask()
   {
       SessionId = "MySessionTest"
   };

   gameInvite.Completed += new
EventHandler<TaskEventArgs>(gameInvite_Completed)
;
   gameInvite.Show();
}

void gameInvite_Completed(object sender,
TaskEventArgs e)
{
   if (e.TaskResult == TaskResult.OK)
       MessageBox.Show("Invitation envoyée");
   else
       MessageBox.Show("Erreur lors de l'envoi");
}
```

Une fois la tâche exécutée, l'écran d'invitation s'affiche. Il faut alors renseigner l'adresse mail ou le gamertag du contact que l'on souhaite inviter.




Un exemple d'utilisation :

Bien que cette tâche permette d'inviter un contact afin de jouer, elle peut être utilisée dans n'importe quel type d'application. On peut ainsi s'en servir pour faire de la promotion par exemple.

### 2.6. SaveContactTask

Avec Mango, il est dorénavant possible d'accéder aux contacts de votre Windows Phone depuis n'importe quelle application (Lien 84). Il est également possible d'ajouter un contact à votre carnet de contacts grâce à la tâche SaveContactTask. Pour ce faire, il est nécessaire de renseigner manuellement l'ensemble des informations concernant le contact à ajouter (noms, téléphone, mail…).

La classe SaveContactTask :

**SaveContactTask**

| | |
|---|---|
| Completed | Se déclenche après l'ajout du contact |
| Compagny | Nom de l'entreprise du contact |
| FirstName | Prénom du contact |
| HomeAddressCity | Ville du contact |
| HomeAddressCountry | Pays du contact |
| HomeAddressState | Etat du contact |
| HomeAddressStreet | Adresse du contact |
| HomeAddressZipCode | Code postal du contact |
| HomePhone | Téléphone fixe du contact |
| JobTitle | Travail du contact |
| LastName | Nom du contact |
| MiddleName | Autres prénoms du contact |
| MobilePhone | Téléphone mobile du contact |
| Nickname | Surnom du contact |
| Notes | Remarques sur le contact |
| OtherEmail | Autre mail du contact |
| PersonnalEmail | Mail personnel du contact |
| Suffix | Suffixe du contact |
| Title | Titre du contact |
| Website | Site du contact |
| WorkAddressCity | Ville du travail du contact |
| WorkAddressCountry | Pays du travail du contact |
| WorkAddressState | Etat du travail du contact |
| WorkAddressStreet | Adresse du travail du contact |
| WorkAddressZipCode | Code postal du travail du contact |
| WorkEmail | Mail du travail du contact |
| WorkPhone | Téléphone du travail du contact |
| Show | Exécute la tâche |

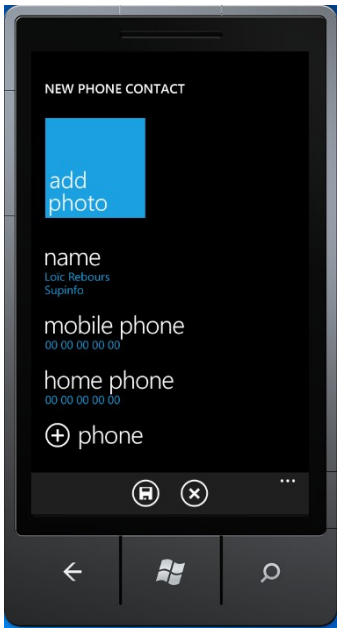
Code :

```
private void SaveContactTaskButton_Click(object
sender, RoutedEventArgs e)
{
   SaveContactTask saveContact = new
SaveContactTask()
   {
       LastName = "Rebours",
       FirstName = "Loïc",
       HomePhone = "00 00 00 00 00",
       MobilePhone = "00 00 00 00 00",
       Website =
"http://www.blog.loicrebours.fr",
       PersonalEmail =
"loic.rebours@supinfo.com",
       Company = "Supinfo",
       HomeAddressCountry = "France",
       HomeAddressCity = "Metz",
       JobTitle = ".NET Developer"
   };

   saveContact.Completed += new
EventHandler<SaveContactResult>(saveContact_Compl
eted);
   saveContact.Show();
}

void saveContact_Completed(object sender,
SaveContactResult e)
{
   if (e.TaskResult == TaskResult.OK)
               MessageBox.Show("Le contact a été
correctement sauvegardé");
       else if (e.TaskResult ==
TaskResult.Cancel)
               MessageBox.Show("La création a été
annulée");
}
```

Une fois la tâche exécutée, le formulaire d'ajout de contact est lancé. Vous pouvez alors choisir le compte (Outlook, Gmail, Live…) sur lequel vous souhaitez ajouter le contact. Vous pouvez également modifier les informations saisies précédemment, puis confirmer ou annuler l'ajout du contact.




Un exemple d'utilisation :

Imaginons une application permettant de mettre en relation une ou plusieurs personnes inconnues. Cette tâche pourrait être utilisée afin d'ajouter en tant que contact cette ou ces nouvelles personnes.

## 2.7. SaveRingtoneTask

SaveRingtoneTask permet d'enregistrer localement une sonnerie qu'il sera alors possible d'utiliser sur votre Windows Phone 7 en tant que sonnerie par défaut.

Pour cela, la musique à enregistrer doit :

- être de type mp3 ou wma ;
- durer moins de 40 s ;
- peser moins de 1Mo ;
- ne pas être protégé par des DRM.

Pour s'exécuter, la tâche a besoin d'un libellé pour la sonnerie, d'un booléen indiquant si la sonnerie en question sera accessible depuis d'autres applications et du chemin – représenté par une URI – vers le fichier en question. Attention, le fichier doit être local pour pouvoir être utilisé par cette tâche. Concrètement, il doit donc être contenu dans le .xap en tant que ressource ou être stocké dans l'isolated storage.

La classe SaveRingtoneTask :

**SaveRingtoneTask**

| | |
|---|---|
| Completed | Se déclenche après la sauvegarde de la sonnerie |
| DisplayName | Nom de la sonnerie |
| IsShareable | Partage avec les autres applications |
| Source | Lien vers le fichier audio |
| Show | Exécute la tâche |

Code :

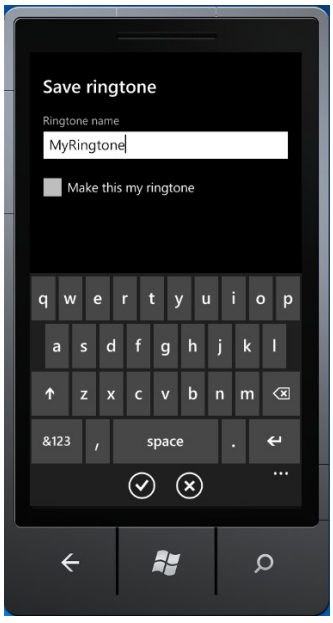
Dans l'exemple ci-dessous, mon MP3 se trouve à la racine de l'application, en tant que ressource.

```
private void SaveRingtoneTaskButton_Click(object
sender, RoutedEventArgs e)
{
   SaveRingtoneTask saveRingTone = new
SaveRingtoneTask()
   {
       DisplayName = "MyRingtone",
       IsShareable = true,
       Source = new Uri("appdata:/starway.mp3")
   };

   saveRingTone.Completed += new
EventHandler<TaskEventArgs>(saveRingTone_Complete
d);
   saveRingTone.Show();
}

void saveRingTone_Completed(object sender,
TaskEventArgs e)
{
   if (e.TaskResult == TaskResult.OK)
               MessageBox.Show("La sonnerie a été
correctement sauvegardée ");
       else if (e.TaskResult ==
TaskResult.Cancel)
               MessageBox.Show("La sauvegarde a
été annulée");
}
```

Une fois la tâche exécutée, vous pouvez modifier le nom de la sonnerie et la définir en tant que sonnerie par défaut, puis confirmer ou annuler l'ajout. Ceci fait, la sonnerie est dès lors accessible depuis le menu Paramètres > Sonneries + sons > Sonnerie.




Un exemple d'utilisation :

Une application permettant de découvrir des morceaux de musique libres sur Internet pourrait utiliser cette tâche afin d'offrir à l'utilisateur la possibilité d'utiliser le début de la musique en question en tant que sonnerie.

## 2.8. ShareLinkTask

Avant Mango, il était particulièrement pénible de publier des informations sur les réseaux sociaux depuis une application. En effet, il n'existait aucune API officielle et il fallait bien souvent passer des heures et écrire plusieurs dizaines/centaines de lignes de code pour ne publier ne serait-ce qu'un petit statut Facebook. Avec Mango, Microsoft propose deux tâches permettant de réaliser cette action d'une façon incroyablement simple. La première, ShareLinkTask permet de partager un lien sur les réseaux sociaux liés au téléphone de l'utilisateur (Facebook, Twitter…). Pour cela, il est nécessaire de fournir à la tâche un titre pour le lien, un message personnel pour accompagner le lien, ainsi que le lien en question, représenté par une URI.

La classe ShareLinkTask :

**ShareLinkTask**

| | |
|---|---|
| Title | Titre décrivant le lien |
| Message | Message personnel |
| LinkUri | Lien à partager |
| Show | Exécute la tâche |

Code :

```
private void ShareLinkTaskButton_Click(object
sender, RoutedEventArgs e)
{
   ShareLinkTask shareLink = new ShareLinkTask()
       {
           Title = "Winrumors",
           Message = "Windows Phone 7 Marketplace
hits 30,000 applications",
           LinkUri = new
Uri(@"http://www.winrumors.com/windows-phone-7-
marketplace-hits-30000-applications/")
       };

   shareLink.Show();
}
```

Lors de son exécution, la tâche lance l'application permettant de partager du contenu sur les réseaux sociaux. L'utilisateur peut alors modifier le message et sélectionner les réseaux sociaux sur lesquels il souhaite effectuer cette publication (note : cet écran ne s'affiche pas depuis l'émulateur WP7). Voici le résultat observé depuis Facebook et Twitter.


Un exemple d'utilisation :

Une application agissant en tant que lecteur RSS pourrait utiliser cette tâche afin de permettre à ces utilisateurs de partager l'URL d'un flux qu'ils apprécieraient

particulièrement.

### 2.9. ShareStatusTask

À l'instar de la précédente tâche, ShareStatusTask permet de partager un statut sur les réseaux sociaux. Cette fois, il suffit uniquement d'indiquer le statut en question pour lancer la publication.

La classe ShareStatusTask :


Code :

```
private void ShareStatusTaskButton_Click(object
sender, RoutedEventArgs e)
{
   ShareStatusTask shareStatus = new
ShareStatusTask()
   {
       Status = "Hello World !"
   };

   shareStatus.Show();
}
```

Là encore, l'application de partage sur les réseaux sociaux se lance et permet de modifier le message et de sélectionner les réseaux sociaux sur lesquels le statut sera publié (note : cet écran ne s'affiche pas depuis l'émulateur WP7). Voici le résultat depuis Windows Live, Facebook et Twitter.


Un exemple d'utilisation :

Vu la présence des réseaux sociaux aujourd'hui, cette tâche peut être présente dans la plupart des applications.

### 3. Conclusion

Cet article vous a présenté les nouvelles tâches introduites par la dernière version du kit de développement à destination de Windows Phone 7 Mango. Comme je le disais au début, certaines devraient rapidement devenir indispensables, comme ShareStatusTask qui permet de publier de façon très simplifiée sur les réseaux sociaux. D'autres, telles que GameInviteTask, trouveront leur utilisation dans des applications bien plus spécifiques. Quoi qu'il en soit, toutes sont les bienvenues et sont d'une simplicité extrême à utiliser.

Vous pouvez retrouver l'ensemble des exemples ci-dessus dans ce projet : Lien 85.

*Retrouvez l'article de Loïc Rebours en ligne : Lien 86*

## Héritage multiple en C++ - Pourquoi l'héritage d'interfaces est insuffisant

Cet article a pour objectif d'étudier les différents types d'héritage multiple que l'on trouve généralement dans les langages de programmation orientés objet à base de classes. Plus précisément, il va comparer l'héritage multiple à une version bridée, nommée héritage multiple d'interfaces, que l'on trouve dans certains langages ; ceci afin de montrer en quoi les limitations introduites par cette dernière me semblent problématiques pour mettre en place une bonne architecture de code.
Cet article n'est pas une introduction à l'héritage (qu'il soit simple ou multiple). Il n'entrera pas non plus dans les détails et les difficultés d'implémentation de l'héritage multiple mais se concentrera sur le point de vue de l'utilisateur du langage.

### 1. Introduction

Le C++ fournit la notion d'héritage multiple, c'est-à-dire la possibilité pour une classe d'avoir non pas une, mais plusieurs classes de base. Cette fonctionnalité est assez controversée, souvent considérée comme trop complexe et peu utile. Beaucoup de langages ayant leurs origines dans le C++ ont décidé de n'en permettre qu'une version limitée : l'héritage multiple d'interfaces. Ayant dû travailler avec un de ces langages, le C#, j'ai pu constater en pratique combien l'héritage multiple me manquait et en quoi son absence entraînait la duplication de code.

C'est cette expérience que je souhaite partager ici, ainsi que quelques informations sur la manière dont on peut mettre en œuvre l'héritage multiple en C++.

### 2. Héritage simple

Le but de l'héritage est double : permettre à une classe dérivée d'accéder à une implémentation définie dans la classe de base, et permettre de définir une interface commune par laquelle plusieurs classes vont pouvoir être manipulées indifféremment.

On dit que l'héritage est simple quand il permet à une classe de dériver d'une classe et d'une seule. Cela peut sembler une limitation arbitraire et ça l'est. Il y a de nombreux cas où le design demanderait que l'on hérite de plusieurs classes. Le problème est que techniquement, un tel héritage est significativement plus compliqué à mettre en œuvre. Il peut être aussi plus complexe à utiliser pour le développeur. Ces deux points conjugués font que de nombreux langages orientés objet ont décidé de ne pas l'autoriser.

### 3. Une version bridée : l'héritage d'interfaces

La plupart des langages ne fournissant pas d'héritage multiple en proposent néanmoins une version limitée, nommée héritage d'interfaces. C'est-à-dire qu'en plus de définir des classes, il est possible dans ces langages de définir des interfaces : des classes n'ayant pas de données membres et n'ayant que des fonctions virtuelles pures (Des fonctions *virtuelles* sont des fonctions pouvant être redéfinies dans les classes dérivées. Dans certains langages, comme Java, toutes les fonctions sont virtuelles par défaut. Des fonctions *virtuelles pures* sont des fonctions virtuelles n'ayant pas d'implémentation dans la classe de base. Le terme *abstraite* est parfois utilisé pour les décrire). Une classe peut alors hériter d'une classe de base unique et d'un nombre quelconque d'interfaces.

Comme une interface ne définit aucune implémentation, hériter d'une interface ne permet donc de réaliser qu'un seul des deux objectifs de l'héritage : manipuler des classes différentes par l'intermédiaire de leur interface, mais pas de bénéficier de l'implémentation d'une classe de base.

Par rapport à l'héritage multiple, cette solution est plus simple à comprendre et à implémenter (même si son implémentation demande quand même pas mal d'efforts et reste relativement coûteuse à l'exécution). Elle est aussi moins puissante. Si ce manque de puissance n'était visible que dans des cas rares, on pourrait s'en accommoder, mais il y a au moins trois cas pour lesquels on se heurte avec désagrément à ces limitations :

- une interface ne permet pas de mettre en place des vérifications liées à la programmation par contrat ;
- hériter de l'implémentation peut être utile ;
- l'héritage multiple permet aussi de définir des types utilisant des politiques et dont l'interface varie.

#### 3.1. Programmation par contrat

Dans la programmation par contrat, l'utilisateur d'une fonction a signé un contrat avec l'implémenteur de cette fonction. Ce contrat se décrit généralement sous la forme de préconditions qui doivent être vraies quand la fonction est appelée, et de postconditions qui doivent être vraies une fois que la fonction retourne.

On vérifie généralement ces conditions par des instructions du type "assert", qui lèvent une erreur à l'exécution si une condition n'est pas vérifiée et qui sont désactivables dans les cas où l'on a besoin de performances maximales.

Quand on parle programmation par contrat et héritage, le

respect du LSP (Liskov Substitution Principle : Lien 87) peut s'énoncer ainsi :

- dans une classe dérivée, on a uniquement le droit d'avoir des préconditions moins strictes que dans la classe de base ;
- dans une classe dérivée, on a uniquement le droit d'avoir des postconditions plus strictes que dans la classe de base.

Dit autrement, la classe dérivée doit au moins fournir les mêmes services que la classe de base. Souvent, on peut se contenter, en termes de validation, d'une forme réduite de ces règles (mais le mécanisme que je vais présenter permet d'implémenter les règles dans leur forme générique, c'est juste plus long à expliquer) : une classe dérivée doit avoir des fonctions avec les mêmes préconditions et postconditions que dans la classe de base.

Comment concrètement mettre en œuvre la vérification de ces conditions ? Il faut les vérifier à chaque appel de la fonction par le client, et ce quelle que soit la manière dont une classe dérivée est implémentée. On voit tout de suite qu'écrire du code ainsi pose problème :

```
class CompteEnBanque
{
public:
 int solde();
 virtual void retirer(int montant)
 {
   // préconditions
   assert(montant > 0);
   assert(montant > solde());
   int monAncienSolde = solde();
   // Réaliser le transfert
   // postconditions
   assert(solde() == monAncienSolde - montant);
 }
};
```

En effet, une personne qui dériverait de la classe CompteEnBanque aurait tout loisir de ne pas respecter le contrat, puisqu'il redéfinit la fonction "retirer". Une autre façon de voir les choses est de dire qu'actuellement, la fonction "retirer" fait deux choses : elle assure l'interface avec l'utilisateur de la classe (et donc le respect du contrat) et elle sert de point d'entrée pour une personne dérivant de cette même classe. Une fonction qui fait deux choses en fait une de trop.

La solution classique, parfois nommée NVI (non virtual interface : Lien 88) est donc la suivante :

```
class CompteEnBanque
{
public:
 int solde();
 void retirer(int montant)
 {
   // préconditions
   assert(montant > 0);
   assert(montant > solde());
   int monAncienSolde = solde();
   doRetirer(montant);
   // postconditions
   assert(solde() == monAncienSolde - montant);
 }
private:
 virtual void doRetirer(int montant) = 0;
};
```

Comme la fonction publique "retirer" est non virtuelle, tout utilisateur manipulant un compte en banque par cette classe de base va passer par ce code et donc voir son contrat respecté, quelle que soit la manière dont on a pu dériver de cette classe.

Dit autrement, toute fonction virtuelle devrait être privée (ou protégée, si on estime que les classes dérivées ont de bonnes raisons d'utiliser l'implémentation de la classe de base) et toute fonction publique devrait vérifier les conditions.

Or, dans les langages présentant ce concept (en C# par exemple), toutes les fonctions définissant une *interface* sont obligatoirement virtuelles et publiques. Elles ne peuvent pas non plus contenir de code de validation (ou d'instrumentation, log ou mesure de performances, par exemple). Et même, il est parfois impossible avec ces langages d'avoir une fonction virtuelle et privée, c'est-à-dire que contrairement au C++, les droits d'accès ne servent pas uniquement à indiquer qui aura le droit d'appeler une fonction, mais aussi qui aura de droit de la redéfinir. Ces langages ne se prêtent donc pas nativement à la vérification de conditions pour la programmation par contrat.

Il existe néanmoins depuis peu un mécanisme facilitant la programmation par contrat en C#, mais il a lieu en dehors du langage et passe par des outils modifiant le comportement du compilateur. Pour les aspects de pré et post conditions liés aux interfaces ou aux fonctions virtuelles, il consiste à écrire des instructions qui vont demander au compilateur d'ajouter du code dans toutes les classes implémentant l'interface.

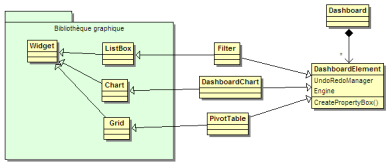
### 3.2. Héritage d'implémentation

Un autre cas où l'héritage multiple non bridé est souvent utile est quand on veut s'insérer dans une hiérarchie existante, tout en apportant sa propre structure complémentaire. Par exemple, imaginons que nous développions une interface graphique permettant d'afficher un tableau de bord de gestion.

Ce tableau est composé d'éléments à placer dans des cases. Chaque élément possède des points communs. Par exemple, il est associé à une boîte de propriétés qui permet de le configurer, il a un point d'accès à un moteur de calculs d'où il peut tirer ses données à afficher et est relié à un gestionnaire d'undo/redo.

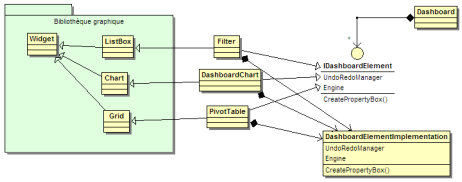
Mais chaque élément est aussi un élément graphique, au sens de la bibliothèque d'IHM qui va être utilisée pour gérer le projet et cette bibliothèque, fournie par une tierce partie, n'est bien entendu pas modifiable. Selon la nature de ce qu'on veut afficher, il peut être judicieux de dériver d'un composant affichant une liste d'objets, d'une grille, d'un composant affichant des camemberts...

Avec de l'héritage multiple, le design ressemblerait à celui-ci :




Comment approcher cette situation avec simplement un héritage d'interfaces ? Si on avait la possibilité de modifier le code de la bibliothèque, on pourrait réviser totalement la hiérarchie de manière à ce que les classes comme ListBox dérivent de DashBoardElement, mais ce serait un peu bancal, toutes les listes déroulantes n'ayant pas pour vocation d'être un élément de tableau de bord.

On pourrait faire en sorte que DashBoardElement ne soit qu'une interface. Le problème est alors que l'on doit dans chaque classe implémentant cette interface dupliquer le code lié à celle-ci, généralement identique d'une classe à l'autre (on peut noter que dans le design initial, aucune des fonctions de DashBoardElement n'était virtuelle). Si ce code est assez gros, on peut en déléguer l'implémentation à une autre classe, mais on doit quand même dupliquer l'appel à cette implémentation séparée. C'est ce que recommandent les gens qui disent que l'héritage multiple peut avantageusement être remplacé par de la composition :




On va quand même dans chaque classe implémenter des fonctions qui ne font que transmettre à la classe d'implémentation :

```
void PivotTable::SetEngine(Engine e)
{
 myImplementation.SetEngine();
}
```

Déjà que cette écriture est fastidieuse, et donc source de bogues d'inattention, elle présente un autre problème : à chaque fois qu'il faut modifier l'interface, il faut reprendre toutes les classes qui implémentent cette interface afin d'y répercuter la modification, alors que dans le design initial, seules les classes directement concernées par la modification devaient en tenir compte.

On a donc au final un design plus complexe, avec des recopies de code en grand nombre et une moindre souplesse.

### 3.3. Mise en place de politiques

Imaginons un type que l'on veut rendre paramétrable de façon à ce qu'il puisse répondre à divers besoins. Si ce paramétrage est constant et peut être connu à la compilation, l'idéal pour faire cela en C++ est d'utiliser les templates. Prenons par exemple la définition d'une classe d'arbre binaire trié configurable.

Cet arbre peut être configuré de multiples manières : comment les nœuds sont comparés entre eux, comment l'arbre est rééquilibré, comment sont alloués les nœuds et bien entendu le type de données stockées dans l'arbre. Cette classe pourrait être définie ainsi :

```
template <class T, class Comparator, template
<class> class AllocationPolicy, class
BalancingPolicy> class Tree;
```

Et l'utilisateur pourrait instancier selon ses besoins un :

```
Tree<string, less<string>, CreateWithNew,
RedBlack> monArbre;
Tree<NobelPrice, SortByYearAndName,
CreateInMemoryPool, AVL> monAutreArbre;
```

On peut a priori implémenter cette classe sans héritage, jusqu'à ce qu'une politique ait besoin de fonctions supplémentaires pour être utilisée. Par exemple, imaginons que dans la politique d'allocation par pool mémoire, on veuille pouvoir indiquer à l'arbre dans quel pool la mémoire doit être allouée :

```
template <class Node> class CreateInMemoryPool
{
public:
 Node* Create();
 void SetMemoryPool(MemoryPool<Node> *pool);
 // ...
};
```

On veut donc sur un arbre implémenté à base de CreateInMemoryPool, et sur ce type d'arbre seulement, pouvoir appeler la fonction SetMemoryPool, fonction dont on ne connaissait même pas l'existence au moment où l'on a défini la classe Tree. Tree doit offrir à son utilisateur une interface qui va dépendre des politiques utilisées.

La méthode classique pour y parvenir est de faire hériter la classe Tree de ses politiques (noter le pluriel : une classe a plusieurs politiques) et comme les classes de politique sont par définition des classes avec du code dedans et avec éventuellement des données membres, on a besoin d'un véritable héritage multiple, pas d'un héritage d'interfaces :

```
template <
 class T,
 class Comparator,
 template <class> class AllocationPolicy,
 class BalancingPolicy>
class Tree :
 public Comparator,
 public AllocationPolicy<Node<T>>,
 public BalancingPolicy
{
 // ...
};
```

## 4. Héritage multiple

J'espère vous avoir convaincu dans les chapitres précédents de l'intérêt de l'héritage multiple. Ce chapitre revient maintenant sur certains points de détail à maîtriser pour utiliser correctement ce concept. Si l'implémentation

peut être assez complexe, l'utilisation de cette fonctionnalité n'est, dans la plupart des cas, pas très compliquée.

### 4.1. Appels ambigus

Comme on peut avoir plusieurs classes de base, elles peuvent entrer en conflit les unes avec les autres. Que se passe-t-il s'il y a ambiguïté entre deux fonctions ou données membres ?

```
class A
{
public:
 void f();
};

class B
{
public:
 void f();
};

class C : public A, public B
{
public:
 void g() {f();} // ?
}
```

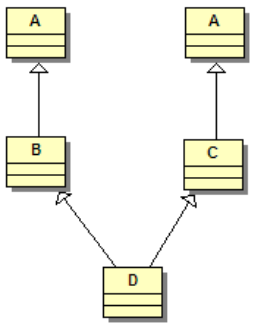
Tout simplement, le compilateur refusera de compiler ce code ambigu et demandera à ce que l'appel soit qualifié : A::f() ou B::f(). À noter que si une fonction f est définie directement au niveau de C, c'est elle qui aura la priorité et il n'y aura pas d'ambiguïté.

### 4.2. Héritage en diamant

Il est interdit d'hériter directement plusieurs fois de la même classe de base.

```
class A;
class B : public A, public A {} // Erreur
```

Par contre, on peut se retrouver à hériter plusieurs fois indirectement de la même classe de base. Si cette classe de base commune n'a pas de données membres et n'a pas de fonction virtuelle qui aurait été redéfinie différemment dans les classes dérivées, il n'y a pas vraiment de questions à se poser. Si elle en a, on peut vouloir deux situations différentes :




La seconde situation se nomme généralement héritage en diamant, à cause de la forme losange prise par le diagramme de classes. Par défaut, si on écrit le code suivant :

```
class A {};
class B : public A {};
class C : public A {};
class D : public B, public C {};
```

On n'obtient pas un héritage en diamant, mais un héritage où la classe A (et ses données membres) se trouve dupliquée.

Pour obtenir un héritage en diamant, il faut utiliser l'héritage virtuel :

```
class A {};
class B : public virtual A {};
class C : public virtual A {};
class D : public B, public C {};
```

On voit là poindre un premier problème : l'héritage en diamant devrait pouvoir être spécifié au niveau de D mais c'est en fait dès le niveau de B et C qu'il doit être indiqué. Un problème annexe existe : comment la partie A de l'objet va-t-elle être construite ?

En effet, en tant que sous-partie de B, elle devrait être construite selon les paramètres définis dans le constructeur de B et en tant que sous-partie de C, en fonction des paramètres définis dans le constructeur de C. Il y a, dès la construction, ambiguïté (qu'il n'y avait pas dans le cas précédant, où deux sous-parties A étaient construites).

Il a donc été décidé de ne choisir ni l'un ni l'autre, mais d'imposer au constructeur de D de spécifier directement comment la partie A de D sera construite. Tout appel au constructeur de A depuis B ou C sera ignoré lors de la création d'un D.

```
D::D() : A(10), B(42), C(12) {}
```

### 4.3. Cast

En présence d'héritage multiple, on peut avoir quelques surprises quand on transtype (cast) des pointeurs sur objet. En effet, un cast est plus qu'une simple réinterprétation d'un pattern de bits, mais une véritable opération qui peut provoquer des décalages.

```
class A
{
public:
 virtual ~A() {} // Pour dynamic_cast
 int i;
};

class B {int j;};
{
public:
 virtual ~B() {} // Pour dynamic_cast
 int j;
};

class C : public A, public B {int k;};

C c;
C* pc = &c;
```

```
B* pb = &c;
A* pa = &c;
```

On pourrait croire dans le code précédent que pa, pb et pc ont la même valeur et pointent sur la même zone mémoire. C'est faux. Le pointeur pa pointe sur la sous-partie de c où est stocké i, pb pointe sur la sous-partie de C où est stocké j. Ces deux adresses sont différentes entre elles.

De même, le code exécuté quand on fait :

```
C* pc2 = dynamic_cast<C*>(pb);
```

est assez complexe et va provoquer, grâce au RTTI, un décalage inverse de pointeur afin que pc2 pointe bien sur l'ensemble de l'objet de type C et non plus sur la sous-partie de C où est stocké j.

Par contre, si on fait :

```
B* pb2 = reinterpret_cast<B*>(&c);
```

On empêche le compilateur de faire les décalages d'adresse nécessaires et pb2 va pointer n'importe où. Encore plus que dans le cas général, éviter à tout prix reinterpret_cast en présence d'héritage multiple.

## 5. Conclusion

On vient de voir que l'héritage multiple peut avoir son lot de complexités. Ces dernières sont de trois natures différentes :

1. plus de travail à faire lors d'un cast de pointeur. Il s'agit là d'une complexité uniquement visible par une personne développant un compilateur C++, sauf si on commence à jouer avec reinterpret_cast, ce qu'on ne devrait de toute façon jamais faire dans ce genre de situation ;
2. il peut y avoir des appels de fonction ambigus, mais ils sont détectés à la compilation et ne devraient pas poser de véritables problèmes pour être corrigés ;
3. l'héritage en diamant apporte à lui seul toute une série de particularités supplémentaires.

L'héritage multiple n'en reste pas moins une technique très appréciable pour mettre en place une architecture propre. On peut décider de se limiter en C++ au sous-ensemble de l'héritage multiple qui reproduit les fonctionnalités fournies par les interfaces de C# ou Java. Un exemple classique est le cas où une hiérarchie d'interface est mise en parallèle d'une hiérarchie d'implémentation, chaque classe d'implémentation devant alors hériter d'une classe d'implémentation et hériter virtuellement de la classe d'interface correspondante, ce qu'on nomme parfois de l'héritage en treillis.

Mais on peut aussi aller plus loin et autoriser les classes de base à contenir d'autres éléments que des fonctions virtuelles pures. Ce qui est notable, c'est que dans les trois exemples cités où l'héritage multiple avec classes de base non creuses avait de l'intérêt, la problématique de l'héritage en diamant, la plus gênante, n'existait pas, on n'avait même pas à se poser la question.

Donc, même si on peut avoir peur des difficultés à maîtriser l'héritage multiple à partir du moment où une même classe de base intervient plusieurs fois, ce qui n'est pas forcément si courant en pratique et il serait dommage de se passer entièrement de la souplesse apportée par l'héritage multiple juste à cause de la crainte de se retrouver dans une situation d'héritage en diamant.

*Retrouvez l'article de Loïc Joly en ligne : Lien 89*

# Qt

## Les dernières news

### Le Qt Project est là

#### Le projet d'open gouvernance pour le framework C++ est arrivé à terme

Une nouvelle page s'écrit dans l'histoire de Qt : l'open gouvernance est arrivée, après de longs mois de préparation.

Tout le développement de Qt aura maintenant lieu sur qt-project.org (Lien 90), le nouveau lieu central, qui héberge un wiki contenant toute la documentation pour commencer à contribuer, comprendre les nouveaux principes en jeu.

Les patches seront gérés par un serveur Gerrit, où des *Approvers* et *Maintainers* pourront les accepter et les intégrer dans les sources après contrôle et tests par la communauté. *Contributors*, *Approvers* et *Maintainers* ne sont plus forcément des employés de Nokia (Thiago, *Maintainer* de Qt Core, par exemple, ne l'est pas), ils sont élus par méritocratie, seul son passé dans le projet entre en ligne de compte. Le développement aura donc maintenant lieu en place publique, tous seront sur un pied d'égalité.

Les développeurs de Qt voient dans cette nouvelle page une excellente nouvelle pour le projet, même si certains sceptiques y voient une occasion pour Nokia de se désengager de Qt.

*Commentez la news de Thibaut Cuvelier en ligne : Lien 91*

## Les dernier tutoriels et articles

### Le livre blanc Qt

Ce livre blanc décrit le framework C++ Qt. Qt aide au développement d'interfaces graphiques multiplateformes avec son approche « écrit une fois, compilé n'importe où ». Utilisant une seule arborescence de source et une simple recompilation, les applications peuvent être exécutées sous Windows, Mac OS X, Linux, Solaris, HP-UX et bien d'autres versions d'Unix avec X11. Les applications Qt peuvent également être compilées pour s'exécuter sur des plateformes embarquées comme Linux embarqué, Symbian ou Windows CE. Qt fournit un excellent support multiplateforme pour le multimédia et les représentations 3D, l'internationalisation, SQL, XML et les tests unitaires, tout en proposant des extensions spécifiques aux plateformes pour les applications spécialisées.

**1. L'article original**

Le Qt Developer Network est un réseau de développeurs utilisant Qt afin de partager leur savoir sur ce framework. Vous pouvez le consulter en anglais : Lien 92.



Cet article est la traduction de Qt Whitepaper : Lien 93.

**2. Introduction**

Les applications Qt peuvent être conçues de manière visuelle en utilisant Qt Designer, un constructeur flexible d'interfaces utilisateurs qui peut être intégré aux EDI.

La version 4.7 de Qt introduit les fondements de Qt Quick, une série de technologies pour un prototypage rapide et la création d'interfaces utilisateurs modernes et intuitives. Elles sont abordées brièvement dans la section Qt Quick plus loin dans ce livre blanc et sont décrites plus en détail dans le livre blanc *Introduction à Qt Quick pour les développeurs C++*.

*Qt est, de fait, le framework C++ standard pour le développement de logiciels multiplateformes de haute performance. En plus d'une bibliothèque de classes complète, Qt inclut des outils qui rendent l'écriture des applications plus rapide et claire. Les capacités multiplateformes et l'aide en termes d'internationalisation assurent aux applications Qt de toucher le marché le plus large possible.*

Le framework C++ Qt est au cœur du développement d'applications commerciales depuis 1995. Qt est utilisé par des compagnies et des organisations aussi diversifiées que Adobe®, Boeing®, Google®, IBM®, Motorola®, NASA, Skype® et par un grand nombre de compagnies et d'organisations plus petites. Qt 4 est conçu pour être utilisé plus facilement que la précédente version de Qt, tout en ajoutant des fonctionnalités plus puissantes. Les classes de Qt sont multifonctions et proposent des interfaces cohérentes pour aider à l'apprentissage, réduire la charge de travail du développeur et augmenter sa productivité. Qt est, et a toujours été, complètement orienté objet.

Ce livre blanc donne un aperçu des outils et des fonctionnalités de Qt. Chaque section débute avec une introduction non technique avant de fournir une description plus détaillée des fonctions pertinentes. Des liens vers les ressources en ligne sont également mis à disposition pour chaque section.

Pour évaluer la version commerciale de Qt pendant 30

jours, voir le site de l'éditeur (Qt est aussi disponible gratuitement sous les termes de la LGPL 2.1) : Lien 94.

### 2.1. Résumé des points importants

Qt inclut une large panoplie de widgets (contrôles dans la terminologie Windows) qui assurent les fonctionnalités standards de l'interface graphique. Qt introduit une alternative innovante pour la communication interobjet, également connue sous le nom de « système de signaux et de slots », qui remplace l'ancienne fonction de rappel non sécurisée utilisée dans de nombreux frameworks historiques. Qt propose également un modèle d'événements conventionnels pour gérer les clics de souris, les pressions des touches du clavier et d'autres interactions avec l'utilisateur. Les applications multiplateformes d'interfaces graphiques avec Qt peuvent gérer toutes les fonctionnalités de l'interface utilisateur requises par une application moderne, comme les menus, les menus contextuels, le glisser-déposer et les barres d'outils de type dock. Les fonctions d'intégration au bureau fournies par Qt peuvent être utilisées pour étendre les applications dans l'environnement du bureau, tirant avantage de certains services disponibles sur chacune des plateformes.

Qt contient également Qt Designer, un outil pour dessiner graphiquement des interfaces utilisateurs. Qt Designer reprend les puissantes fonctionnalités en termes de disposition de Qt en plus d'une traduction absolue. Qt Designer peut être utilisé simplement pour la conception des interfaces graphiques, ou pour créer des applications entières avec sa capacité à s'intégrer dans des environnements de développement intégrés (EDI) populaires.

Qt est une excellente aide pour le multimédia et les représentations 3D (voir plus loin). Qt est, de fait, le framework d'interfaces graphiques standard pour la programmation indépendamment de la plateforme OpenGL®. Le système de dessin de Qt offre un rendu de haute qualité à travers toutes les plateformes supportées. Un canevas de framework sophistiqué permet aux développeurs de créer des applications graphiques interactives qui tirent avantage des fonctionnalités avancées de dessin de Qt.

Qt rend possible la création d'applications basées sur des données indépendantes de la plateforme en utilisant des bases de données standard. Qt contient des pilotes natifs pour Oracle®, Microsoft®, SQL Server, Sybase® Adaptative Server, IBM DB2, PostgreSQL™, MySQL®, Borland® Interbase, SQLite et les bases de données accessibles par ODBC. Qt contient des widgets spécifiques aux bases de données et tous les widgets intégrés et personnalisés peuvent s'adapter aux données.

Les programmes Qt ont une ergonomie native sous toutes les plateformes supportées utilisant les styles et les thèmes acceptés par Qt. À partir d'une seule arborescence de sources, la recompilation est la seule chose requise pour produire des applications pour Windows® XP®, Windows Vista™ et Windows 7™, Mac OS X®, Linux®, Solaris™, HP-UX™ et bien d'autres versions d'Unix® avec X11. L'outil de compilation de Qt qmake produit des makefiles ou des fichiers .dsp selon la nature de la plateforme cible.

Comme l'architecture Qt tire avantage de la plateforme sous-jacente, de nombreux clients utilisent Qt pour un développement mono-plateforme sous Windows, Mac OS X et Unix car ils préfèrent l'approche de Qt. Qt inclut une aide pour d'importantes fonctions spécifiques aux plateformes, comme ActiveX® sous Windows et Motif™ sous Unix. Voir la section Architecture de Qt. pour plus d'information : Lien 95.

Qt utilise dans son ensemble l'Unicode™ et fournit un support considérable pour l'internationalisation. Qt contient Qt Linguist et d'autres outils pour aider les traducteurs. Les applications peuvent facilement utiliser et mélanger l'arabe, le chinois, l'anglais, l'hébreu, le japonais, le russe et d'autres langues définies par l'Unicode.

Qt contient une variété de classes dans des domaines spécifiques. Par exemple, Qt possède un module XML qui inclut des classes SAX et DOM pour lire et manipuler des données stockées en XML. Les objets peuvent être sauvegardés en mémoire en utilisant les classes de collections compatibles avec STL de Qt et manipulés en utilisant les styles d'itérateurs Java® ainsi que la bibliothèque standard C++ (STL). La manipulation des fichiers locaux et distants utilisant des protocoles standards est fournie par les classes Qt d'entrée/sortie et de réseau.

Les applications Qt peuvent voir leurs fonctionnalités étendues par des plug-ins et des bibliothèques dynamiques. Les plugins fournissent des codecs supplémentaires, des pilotes de bases de données, des formats d'images, des styles, et des widgets. Les plug-ins et les bibliothèques peuvent être vendus en tant que produits indépendants.

Le module QtScript permet d'écrire des applications avec Qt Script, un langage de script basé sur ECMAScript, lié à JavaScript. Cette technologie permet aux développeurs de définir des restrictions d'accès pour les utilisateurs pour certaines parties de leurs applications en ce qui concerne l'écriture des scripts.

Qt est un framework mature pour C++ qui est largement utilisé de par le monde. En plus des nombreuses utilisations commerciales de Qt, l'édition Open Source de Qt constitue les fondations de KDE, l'environnement de bureau de Linux. Développer des applications devient un plaisir grâce à Qt, avec son système de construction multiplateforme, la conception graphique des pages et une API élégante.

### 2.2. Référence en ligne

Qt in Use : Lien 96.

## 3. Interfaces graphiques utilisateur

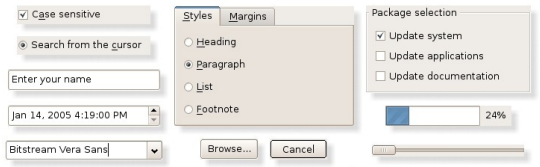
Qt offre un ensemble très riche de widgets standard utilisables pour créer des applications composées d'interfaces graphiques. Des gestionnaires de disposition sont utilisés pour organiser et redimensionner les widgets afin de les adapter à l'écran, à la langue et à la police de l'utilisateur.

Les widgets sont des éléments visuels qui sont combinés

pour créer des interfaces utilisateur. Les boutons, les menus et les barres de défilement, les boîtes de dialogue et les fenêtres d'application sont tous des exemples de widgets.

Les gestionnaires de disposition organisent les widgets enfants à l'intérieur de la zone de leur widget parent. Ils effectuent le positionnement et le redimensionnement automatique des widgets enfants, donnent aux widgets de haut niveau des dimensions judicieuses minimales et par défaut et repositionnent automatiquement les widgets lorsque leur contenu change.

Les signaux et slots connectent les composants de l'application les uns avec les autres pour qu'ils communiquent d'une façon simple en assurant la sûreté du typage. Cette forme de communication interobjet est disponible dans tous les widgets standard et peut être utilisée par les développeurs dans leurs propres widgets personnalisés.




Une sélection de widgets disponibles avec Qt.

## 3.1. Widgets

Les images ci-dessus présentent une sélection de widgets. Ceci inclut des widgets standards de saisie de texte, des cases à cocher, des boutons radio, des curseurs et des boutons d'action, ainsi que des widgets plus spécialisés pour la saisie de dates ou d'horaires.

Les libellés, fenêtres de message, infobulles et autres widgets textuels ne sont pas limités à l'usage d'une seule couleur, police ou langue. Les widgets d'interprétation textuelle peuvent afficher du texte enrichi multilingue en utilisant un sous-ensemble de HTML.

Les widgets conteneurs comme les widgets d'onglets et les groupes de contrôles sont également disponibles et peuvent être utilisés pour regrouper les composants liés de l'interface utilisateur. Ces widgets sont gérés spécifiquement dans Qt Designer pour aider les concepteurs d'interfaces à créer de nouvelles interfaces utilisateur. Les widgets plus complexes, comme les vues déroulantes, sont souvent utilisés plutôt par les développeurs que par les concepteurs d'interfaces utilisateur, car ils sont utilisés pour afficher du contenu spécialisé ou dynamique.

Les développeurs peuvent créer leurs propres widgets et fenêtres de dialogue en héritant de la classe de base QWidget ou de l'une de ses sous-classes. Les widgets spécialisés de ce type peuvent être complètement personnalisés pour présenter leur propre contenu, répondre aux actions de l'utilisateur et produire leurs propres signaux et slots.

Qt fournit bien d'autres widgets qui s'additionnent à ceux présentés ici. La plupart des widgets disponibles se présentent avec des liens vers la documentation de leur classe dans la galerie des widgets de Qt en ligne : Lien 97.

## 3.2. Gestionnaires de disposition (layouts)

Les Gestionnaires de disposition procurent une flexibilité et une réactivité aux interfaces utilisateur, leur permettant de s'adapter lors d'un changement de style, d'orientation ou de police de caractères.

Ils aident les développeurs à maintenir l'internationalisation de leurs applications. Avec des positions et des largeurs fixes, un texte traduit est souvent tronqué ; avec eux, les widgets enfants sont redimensionnés automatiquement. De plus, le placement d'un widget peut être inversé pour produire une apparence plus naturelle pour les utilisateurs qui travaillent avec un système d'écriture de droite à gauche.

Les dispositions peuvent également aller de droite à gauche et de bas en haut. Celles de droite à gauche sont commodes pour les applications internationalisées qui supportent les systèmes d'écriture de droite à gauche comme l'arabe ou l'hébreu. Elles sont intégrées complètement dans le système de style de Qt pour procurer une vision pratique et agréable sur les affichages inversés.

Qt Designer est capable d'utiliser les layouts pour les widgets de positionnement.

## 3.3. Signaux et slots

Les widgets émettent des signaux quand des événements ont lieu. Par exemple, un bouton émet le signal clicked lorsqu'il est cliqué. Un développeur peut choisir de connecter un signal en créant une fonction (un « *slot* ») et en appelant la fonction connect() pour rapprocher le signal du slot. Les mécanismes des signaux et des slots de Qt ne nécessitent pas que les classes aient connaissance les unes des autres, ce qui facilite le développement de classes hautement réutilisables. Puisque les signaux et les slots ont un typage sécurisé, les erreurs de type sont reportées en tant qu'avertissements et ne provoquent pas l'apparition de plantages.

Par exemple, si le signal clicked() d'un bouton de fermeture est connecté au slot quit() de l'application, le clic d'un utilisateur sur Quit termine l'application. En termes de code, cela s'écrit comme cela :

```
connect(button, SIGNAL(clicked()), qApp,
SLOT(quit()));
```

Les connexions peuvent être ajoutées ou supprimées à n'importe quel moment pendant l'exécution d'une application Qt, elles peuvent être paramétrées pour s'exécuter lorsqu'un signal est émis ou mises en file d'attente pour une exécution future, elles peuvent aussi impliquer des objets de processus différents.

Le mécanisme des signaux et des slots est implémenté en C++ standard. L'implémentation utilise le préprocesseur C++ et moc, le compilateur de métaobjets, inclus dans Qt. La génération de code est effectuée automatiquement par le système de compilation de Qt. Les développeurs n'ont à aucun moment besoin d'éditer ou même de visualiser le

code généré.

En plus de manipuler les signaux et les slots, le compilateur de métaobjets supporte les mécanismes de traduction de Qt, son système de propriétés et ses informations de type disponibles à l'exécution. Cela permet également une introspection des programmes C++ lors de l'exécution d'une manière qui fonctionne sur toutes les plateformes supportées. Le système sous-jacent qui procure ces capacités est connu sous le nom de système de métaobjets de Qt.

### 3.4. Références en ligne

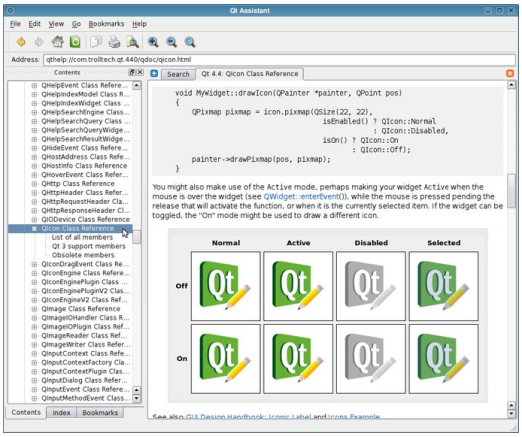
- Exemples : Lien 98.
- Les dispositions : Lien 99.
- Le modèle objet : Lien 100.
- Les signaux et slots : Lien 101.

## 4. Fonctionnalités de l'application

*Construire des applications avec des interfaces graphiques modernes avec Qt est rapide et simple et peut se faire par codage manuel ou en utilisant Qt Designer, l'outil de conception visuel de Qt.*

Qt offre toutes les fonctions nécessaires pour créer des applications avec des interfaces graphiques modernes comprenant menus, barres d'outils et dock Windows. Qt supporte à la fois les SDI (interface à document simple) mais aussi les MDI (interface à documents multiples). Qt supporte également le glisser/déplacer et le presse-papier.

Un ensemble complet de boîtes de dialogue standard sont mises à disposition, incluant celles pour choisir les fichiers, les répertoires, les polices de caractères et les couleurs. En pratique, une instruction sur une ligne constitue la seule chose nécessaire à la présentation d'une boîte de dialogue standard.




Qt Assistant utilise de nombreuses fonctionnalités de la fenêtre principale de l'application pour afficher la documentation de Qt.

Qt utilise les actions pour simplifier la programmation des interfaces utilisateur. Par exemple, si un menu d'options, un bouton de barre d'outils et un accélérateur de clavier produisent la même action, l'action en question ne devra être codée qu'une seule fois.

Qt peut sauvegarder les paramètres de l'application d'une façon indépendante de la plateforme, en utilisant le système de registres ou des fichiers texte, permettant à des objets comme les préférences utilisateur, les fichiers récemment utilisés, les positions et tailles des fenêtres et des barres d'outils d'être sauvegardés pour utilisation future.

La possibilité de programmer en multiprocessus est offerte par une collection de classes qui présentent des constructions communes, permettant d'écrire des applications Qt qui tirent avantage des processus pour effectuer des calculs, des tâches de longue durée ou seulement pour augmenter la réactivité.

Les applications peuvent également utiliser les fonctionnalités d'intégration du bureau Qt pour interagir avec les services fournis par l'environnement de bureau de l'utilisateur.

### 4.1. Fonctionnalités de la fenêtre principale

La classe **QMainWindow** offre un framework pour les fenêtres principales des applications usuelles. Une fenêtre principale contient un ensemble de widgets standard. Le haut de la fenêtre principale est occupé par une barre de menu, sous laquelle les barres d'outils sont disposées dans des zones de barre d'outils autour du centre de la fenêtre. La zone de la fenêtre principale située sous la barre d'outils du bas est occupée par une barre d'état. Les messages « Qu'est-ce que c'est ? » et les infobulles constituent les éléments de l'aide dans l'interface utilisateur.

Le widget **QMenu** présente les objets du menu à l'utilisateur sous forme d'une liste verticale. Les menus peuvent être isolés (par exemple, un menu contextuel sous forme de pop-up), apparaître dans une barre de menu ou être des sous-menus d'un autre menu pop-up. Les menus peuvent avoir des poignées de redimensionnement.

Chaque objet de menu peut avoir une icône, une case à cocher et un accélérateur. Les objets de menu correspondent habituellement à des actions (comme « Sauvegarder ») et provoquent l'exécution des slots qui leur sont associés lorsqu'ils sont sélectionnés par l'utilisateur. Le gestionnaire de layout de Qt prend en considération chaque barre de menu. Sur Mac OS X, la barre de menu apparaît en haut de l'écran.

Les menus de Qt sont très flexibles et font partie d'un système d'actions intégré (voir Actions). Les actions peuvent être activées ou désactivées, ajoutées dynamiquement aux menus et supprimées à nouveau plus tard.


Le support de barre d'outils unifié sur Mac OS X améliore l'aspect visuel et rend agréable l'utilisation des applications en fusionnant les barres d'outils adjacentes et les barres de titre des fenêtres.

Les barres d'outils contiennent des ensembles de boutons ainsi que d'autres widgets auxquels l'utilisateur peut accéder pour effectuer des actions. Elles peuvent être déplacées au choix entre le haut, la gauche, la droite et le

bas de la zone centrale de la fenêtre principale. Chaque barre d'outils peut être glissée hors de sa zone de barre d'outils et flotter comme une palette d'outils indépendante.

La classe **QToolButton** implémente un bouton de barre d'outils avec une icône, un cadre stylisé et un libellé optionnel. Les boutons de basculement de la barre d'outils activent les fonctionnalités ou les désactivent. D'autres boutons de la barre d'outils exécutent des commandes. Différentes icônes peuvent être fournies pour les modes actif, valide et invalide et les états démarré et arrêté. Si une seule icône est fournie, Qt distingue automatiquement l'état en utilisant des artifices visuels (par exemple, griser les boutons inactifs). Les boutons de la barre d'outils peuvent également déclencher des menus pop-up.

Les fenêtres dock sont des fenêtres que l'utilisateur peut déplacer à l'intérieur d'une même zone de barre d'outils ou d'une zone de barre d'outils à une autre. L'utilisateur peut détacher une fenêtre dock et la rendre flottante en haut de l'application ou la réduire. Des animations sont utilisées pour glisser doucement les fenêtres dock dans et hors des zones dock.

Les zones dock peuvent également s'emboîter pour empiler les fenêtres dock en plusieurs lignes ou colonnes, les fenêtres dock peuvent être empilées dans des zones partagées - lorsque c'est le cas, les widgets dock sont contenus dans des onglets.

La personnalisation des fenêtres dock est possible également. Elles peuvent être affichées avec une barre de titre verticale et des barres de titres et des contrôles de fenêtres avec un style particulier donné.

Certaines applications, comme Qt Designer et Qt Linguist, utilisant des fenêtres dock de manière intensive, offrent des opérateurs de sauvegarde et restituent la position des fenêtres dock et des barres d'outils, pour que les applications puissent restituer facilement l'environnement de travail préféré de l'utilisateur.

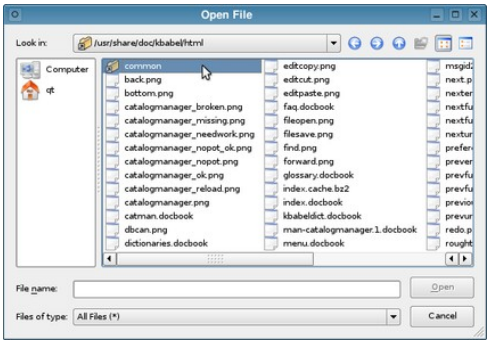
## 4.2. Actions

Les applications offrent habituellement à l'utilisateur la possibilité d'effectuer une action particulière de plusieurs façons bien différentes. Par exemple, la plupart des applications mettent traditionnellement une action « sauvegarder » à disposition via le menu, la barre d'outils (un bouton dans la barre d'outils avec une icône appropriée) et un accélérateur. La classe **QAction** encapsule ce concept. Il permet aux programmeurs de définir une action à un emplacement.

Tout en évitant de faire le même travail plusieurs fois, l'utilisation d'une **QAction** assure que l'état des objets de menu reste en phase avec les boutons de la barre d'outils correspondants et que l'aide interactive soit bien affichée quand cela est nécessaire. Désactiver une action va désactiver tous les objets de menu correspondants et les boutons de la barre d'outils. De même, si l'utilisateur clique sur un bouton de basculement dans la barre d'outils, l'objet de menu correspondant sera basculé lui aussi.

## 4.3. Boîtes de dialogue et assistants

La plupart des applications GUI utilisent des boîtes de dialogue pour interagir avec l'utilisateur pour certaines opérations. Qt inclut des classes de boîtes de dialogue prêtes à l'emploi offrant des fonctions commodes pour les tâches les plus communes. Des captures d'écran de certaines boîtes de dialogue standard de Qt sont présentées ci-dessous. Qt propose également des boîtes de dialogue standard pour la sélection de couleur, l'impression, les indicateurs de progression et les messages d'information.



Une QFileDialog et une QFontDialog présentées dans le style Plastique. Sur Windows et Mac OS X, les boîtes de dialogue natives sont utilisées à la place.

Les programmeurs peuvent créer leurs propres boîtes de dialogue par héritage de la classe **QDialog**. Qt Designer inclut également des modèles de boîtes pour aider les développeurs à démarrer avec la conception.

Les assistants sont utilisés pour guider les utilisateurs à travers les tâches et les procédures communes, les emmenant étape par étape à travers les options disponibles tout en leur proposant de l'aide si nécessaire. Qt offre une API flexible et intuitive pour construire des assistants possédant l'aspect natif approprié et une utilisation conforme aux environnements des plateformes supportées.

La classe **QWizard** offre des fonctionnalités pour personnaliser l'apparence de l'assistant au-delà de l'aspect visuel et ergonomique basique, spécifiquement à la plateforme. Les instances de cette classe contrôlent également l'ordre de présentation des pages à l'utilisateur. **QWizardPage** est un widget standard qui offre des fonctionnalités pour sauvegarder et valider les saisies de l'utilisateur.

## 4.4. Aide interactive

Les applications utilisent souvent des formes variées d'aide interactive pour expliquer le propos d'un élément d'interface et assister l'utilisateur. Qt offre deux mécanismes pour donner des messages d'erreur brefs : les infobulles pour une aide courte liée au contexte et des aides pop-up du style « Qu'est-ce que c'est ? » contenant des messages plus longs et plus informels. Tous deux sont intégrés dans le système d'action Qt.

Les développeurs peuvent déployer l'assistant Qt comme un navigateur d'aide pour leurs propres applications ainsi que l'ensemble de la documentation en utilisant les classes du module d'aide de Qt. Ce module offre également une API que les développeurs peuvent utiliser pour accéder à la documentation dans le but d'obtenir des affichages

personnalisés, peut-être en utilisant les classes infobulles et « Qu'est-ce que c'est ? » pour montrer de petites, mais pertinentes, parties d'information à l'utilisateur.

### 4.5. Paramètres

Les paramètres de l'utilisateur et les paramètres d'autres applications peuvent être sauvegardés facilement en utilisant la classe **QSettings**. Sur les plateformes Windows, Mac OS X et Linux, les paramètres sont sauvegardés dans un système standard de localisation ; pour les autres plateformes, ils sont sauvegardés dans des fichiers texte.

Une variété de types de données Qt peut être utilisée de façon cohérente avec **QSettings**. Ce type est sérialisé pour être stocké et rendu disponible par la suite pour les applications. Voir la gestion des fichiers pour plus d'information à propos de la sérialisation des types de données de Qt.

### 4.6. Programmation multiprocessus et concurrente

Les applications Qt peuvent utiliser de multiples processus : un processus pour garder l'interface utilisateur active et un ou plus pour s'occuper des activités consommatrices de temps, comme la lecture de gros fichiers ou des calculs complexes. Qt propose des classes pour représenter les processus, les exclusions mutuelles, les sémaphores, les stockages de processus globaux et les primitives de verrouillage.

Des facilités pour la programmation concurrente sont également fournies, incluant des implémentations des algorithmes bien connus de map-reduce et filter-reduce. Ils sont intégrés dans le modèle objet de Qt, utilisant des classes conteneurs standard pour rendre plus commode l'utilisation des techniques concurrentes dans les applications Qt.

Le système de métaobjets de Qt active des objets dans les différents processus pour communiquer en utilisant les signaux et les slots, permettant aux développeurs de créer des applications monoprocessus qui pourront plus tard être adaptées pour le multiprocessus, sans pour autant revoir massivement la conception.

### 4.7. Intégration sur le bureau

Les applications peuvent être étendues pour interagir avec des services fournis par l'environnement du bureau de l'utilisateur en utilisant les classes d'intégration du bureau de Qt. Cela va de **QSystemTrayIcon**, qui est souvent utilisée par des applications avec un temps d'exécution long pour garder un indicateur persistant dans la barre de notification, à **QDesktopServices**, qui permet aux ressources comme des URL mailto: et http:// d'être traitée par les applications les plus appropriées sur chaque plateforme.

### 4.8. Référence en ligne

Les classes de Qt pour la fenêtre principale : Lien 102.

## 5. Qt Designer

*Qt Designer est un outil de conception d'interfaces graphiques utilisateur pour les applications Qt. Les applications peuvent être écrites intégralement comme du code source ou en utilisant Qt Designer pour augmenter la vitesse de développement. Une architecture basée sur des composants permet aux développeurs d'étendre Qt Designer avec des widgets et des extensions personnalisées et même de les intégrer dans des environnements de développement intégrés.*

Concevoir un formulaire avec Qt Designer est un processus très simple. Les développeurs glissent les widgets d'une barre d'outils vers le formulaire et utilisent les outils d'édition standard pour les sélectionner, les couper, les coller et les redimensionner. Chaque propriété de widget peut être changée en utilisant l'éditeur de propriétés. Les dimensions et positions précises des widgets importent peu. Les développeurs sélectionnent les widgets et leur appliquent les layouts. Par exemple, certains widgets bouton peuvent être sélectionnés et disposés l'un à côté de l'autre en choisissant l'option « disposition horizontale ». Cette approche rend la conception très rapide et les formulaires définitifs s'adapteront proprement à la dimension de fenêtre que l'utilisateur final préfère. Voir la partie Layouts pour plus d'informations à propos des layouts automatiques de Qt.

Qt Designer supprime le cycle consommateur de temps « compilation, édition des liens et exécution« pour la conception des interfaces utilisateur. Ceci permet de rendre la modification d'un modèle très facile. Les options d'aperçu de Qt Designer permettent à l'utilisateur de voir ses formulaires avec d'autres styles ; par exemple, un développeur Mac OS X peut voir le rendu d'un formulaire dans un style Windows. Les formulaires peuvent être prévisualisés en utilisant le mécanisme de « skins » pour simuler les contraintes graphiques et l'apparence de l'objet cible.

Les licences commerciales de Windows peuvent bénéficier des facilités qu'offre Qt Designer en termes de conception d'interface utilisateur à partir de Microsoft Visual Studio®. Les frameworks de développement Qt proposent également un plug-in d'intégration Qt pour l'EDI multiplateforme Eclipse™, qui embarque Qt Designer tout comme d'autres technologies Qt dans le framework de l'EDI.

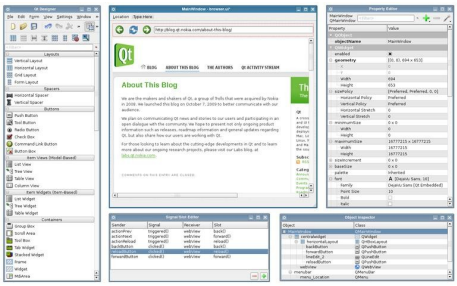
### 5.1. Travailler avec Qt Designer

Les développeurs peuvent créer à la fois des applications du style « boîte de dialogue » ou des applications du style « fenêtre principale » avec des menus, des barres d'outils, des infobulles et d'autres fonctionnalités standard. Plusieurs modèles de formulaires sont proposés et les développeurs peuvent créer leurs propres modèles pour assurer la cohérence dans une même application ou dans une famille d'applications. Les programmeurs peuvent créer leurs propres widgets personnalisés qui peuvent être intégrés simplement avec Qt Designer.

Qt Designer propose une approche basée sur les formulaires pour le développement d'applications. Un formulaire est représenté par un fichier d'interface utilisateur (.ui) qui peut soit être converti en C++ et compilé dans l'application, soit être traité au moment de

l'exécution pour produire des interfaces utilisateur générées dynamiquement. Le système de compilation de Qt est capable d'automatiser la préparation de l'étape de compilation des interfaces utilisateur pour faciliter le processus de conception.

Les outils utilisés pour créer et éditer le code source des applications créées avec Qt Designer vont dépendre des préférences personnelles de chaque développeur ; certains voudront tirer avantage des fonctionnalités d'intégration offertes avec Qt Designer pour développer à partir de Microsoft Visual Studio ou l'environnement Eclipse.



Une vue d'ensemble de l'interface utilisateur de Qt Designer.

## 5.2. Extension de Qt Designer

L'architecture basée sur les composants utilisée comme fondements de Qt Designer a été spécialement conçue pour permettre aux développeurs d'étendre son interface utilisateur et les outils d'édition avec des composants personnalisés. En plus, la nature modulaire de l'application permet de rendre accessibles les fonctionnalités de conception d'interface graphique de Qt Designer à partir des environnements de développement intégrés comme Microsoft Visual Studio et KDevelop.

En tout, le module Qt Designer offre plus de vingt classes pour travailler avec les fichiers .ui et pour étendre Qt Designer. Beaucoup d'entre elles permettent à des tiers de personnaliser l'interface utilisateur de l'application elle-même.

Les tiers et les widgets personnalisés pour un travail interne sont facilement intégrés dans Qt Designer. Adapter un widget existant pour l'utiliser dans Qt Designer nécessite uniquement de compiler le widget en tant que plug-in, en utilisant une classe d'interface pour supporter les propriétés par défaut du widget et construire de nouvelles instances du widget. L'interface du plug-in est exportée vers Qt Designer en utilisant une macro similaire à celle décrite dans la partie Plug-ins.

## 5.3. Références en ligne

- Le manuel de Qt Designer : Lien 103.
- Outils pour développeurs : Lien 104.
- Qt Designer : Lien 105.
- Exemples : Lien 106.

*Retrouvez la suite de l'article du Qt Developer Network traduit par Aldiemus et Thibaut Cuvelier en ligne : Lien 107*

## 4D v12.3 est disponible !

- Certifiée Mac OS X Lion
- Compatible Mac App Store d'Apple
- Support 64 bits
- PHP 5.3 intégré

Retrouvez l'annonce sur le site de 4D : Lien 108
Télécharger : Lien 109

*Commentez la news de Steph4D en ligne : Lien 110*

## Conférence Développeur à Paris

La **Conférence Développeur** à **Paris** aura lieu au **Musée Dapper** le **15 novembre 2011**.

Dédiée à un public de développeurs professionnels, c'est l'occasion de découvrir les produits et leurs nouvelles fonctionnalités, **4D v13** et **Wakanda**, de rencontrer les ingénieurs de 4D, d'échanger avec les développeurs,...

Après un petit-déjeuner d'accueil, vous assisterez aux présentations :

- de la **stratégie 4D-Wakanda**,
- des **nouveautés 4D v13**,
- de la **productivité améliorée** en développant avec 4D v13,
- de la **performance** de vos applications 4D v13,
- de **4D dans le cloud**,
- et de **Wakanda** la nouvelle plate-forme unifiée open-source 100 % JavaScript,

entrecoupé d'un déjeuner et de pauses où vous pourrez discuter avec les ingénieurs et développeurs 4D.

Infos pratiques et agenda complet de la journée : Lien 111

*Commentez la news de Steph4D en ligne : Lien 112*

## Utilisation des variables de contexte avec l'ETL Talend Open Studio

Ce tutoriel va nous permettre de comprendre comment utiliser les variables de contexte avec Talend Open Studio puis comment exploiter ces dernières après l'exportation de notre job sous forme de batch.

### 1. Introduction

L'article suivant nous présente comment utiliser les variables de contexte dans un job assez simple qui lancera une requête "select" sur une base de données Oracle, ensuite nous allons exporter ce job en batch Windows et voir toute l'utilité des variables de contexte.

Avant de commencer je tiens à préciser qu'on n'abordera pas dans ce tutoriel le détail de création de job sous Talend vu que ça a été discuté dans cet autre tutoriel : Création de job avec l'ETL Talend Open Studio (Lien 113).

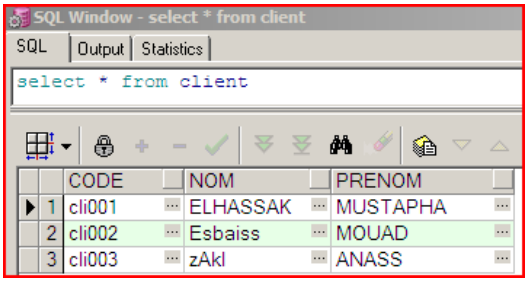
### 2. La source de données

La source de données a été créée par les deux scripts suivants :

```
CREATE TABLE CLIENT(
       CODE VARCHAR2(100 BYTE),
       NOM VARCHAR2(100 BYTE),
       PRENOM VARCHAR2(100 BYTE)
       );
```

```
INSERT INTO CLIENT
VALUES
  ('cli001',
   'ELHASSAK',
   'MUSTAPHA');
INSERT INTO CLIENT
VALUES
  ('cli002',
   'Esbaiss',
   'MOUAD');
INSERT INTO CLIENT
VALUES
  ('cli003',
   'zAkI',
   'ANASS');
```

Finalement on se retrouve avec le résultat suivant :




Le job qu'on va créer exécutera la requête suivante sur notre table, il s'agit d'une sélection de toutes les colonnes de la table avec une condition sur la colonne "CODE", par exemple :

```
SELECT code, nom, prenom
FROM CLIENT
WHERE code LIKE 'cli001'
```

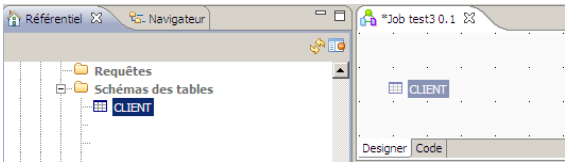
### 3. Job Talend Open Studio

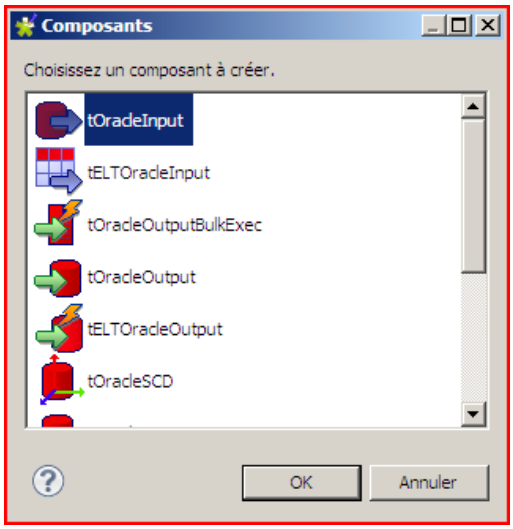
La création du job TOS va se dérouler suivant les étapes :

- création du job ;
- ajout des variables de contexte au job ;
- utilisation des variables de contexte dans les composants ;
- export du job en batch.

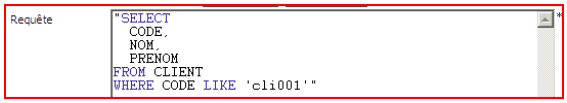
#### 3.1. Création du job

La création du job est assez rudimentaire. Après création d'un job, on glisse le schéma de notre table dans le designer, puis on choisit le composant "tOracleInput".

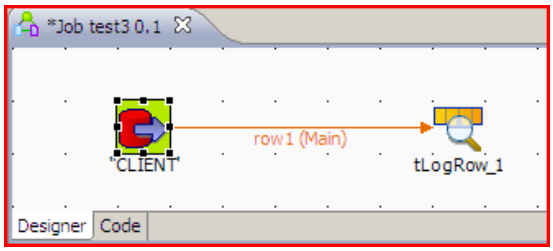


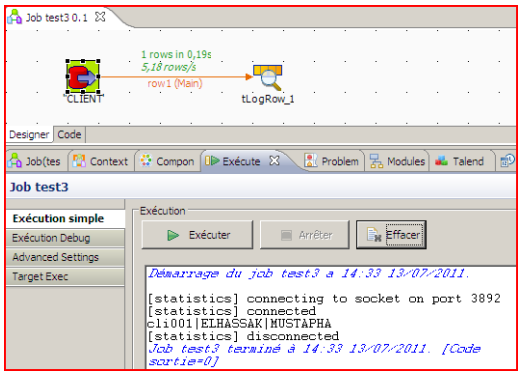

- Après on va sur les propriétés du composant "tOracleInput" et on modifie la requête SQL qu'il utilise.



- Ceci fait, on glisse un deuxième composant depuis la palette, cette fois-ci ce sera un composant "tLogRow" qui servira à afficher le résultat de la requête dans la fenêtre console. Ce composant est utilisé ici juste pour l'affichage. Après on lie ce dernier avec le "tOracleInput" avec un lien de type "main".




- Notre job est fin prêt à l'emploi. On lance l'exécution et voilà le résultat :

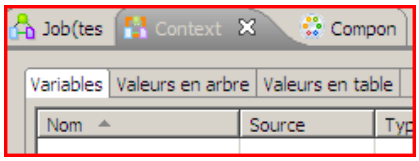



## 3.2. Ajout des variables de contexte au job

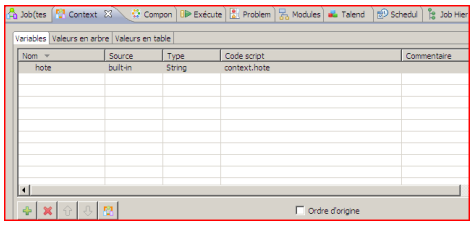
Maintenant que notre job est opérationnel on va commencer à utiliser les variables de contexte, en fait on va essayer de rendre les paramètres de connexion à notre base de données moins statiques, au lieu de les laisser figés dans le job on va les stocker dans des variables de contexte.

Pour ça on commence par créer les variables.

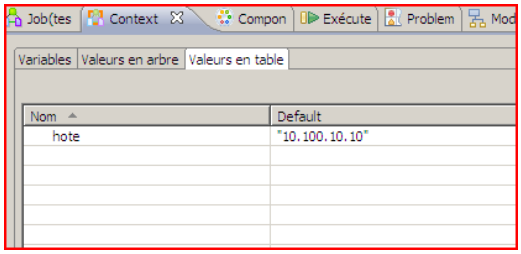
- On commence par cliquer sur l'onglet "Context".



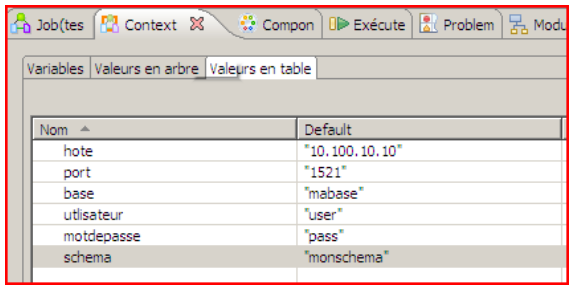

- On clique sur le bouton "+" qui est en bas pour ajouter une variable.
- On lui donne le nom "hote", le type "String" et on laisse les autres colonnes à leur valeur par défaut.



- Enfin pour donner une valeur par défaut à notre variable on va sur l'onglet "Valeurs en table" puis on tape dans la colonne "Default" correspondant à notre variable la valeur qu'on veut.
- Il s'agit d'une variable de type String donc il ne faut pas oublier les guillemets.




- On refait la même chose pour les autres variables.
- Finalement on devrait se retrouver avec quelque chose qui ressemble à ceci :

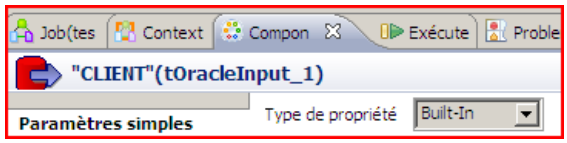



## 3.3. Utilisation des variables de contexte dans les composants

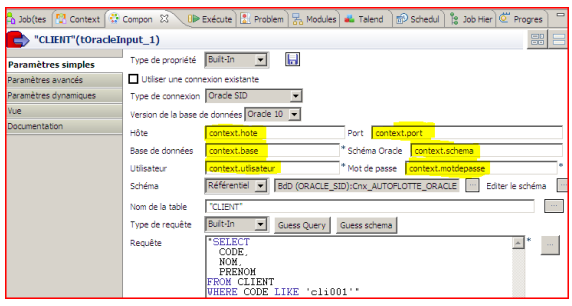
Pour utiliser nos variables précédemment créées il suffit de les appeler n'importe où sous la forme **context.nomDeMaVariable**.

Passons à la pratique, on modifiera les paramètres de notre "tOracleInput".

- On commence par rendre le type de propriété du composant en "Built-in" au lieu de "Référentiel".

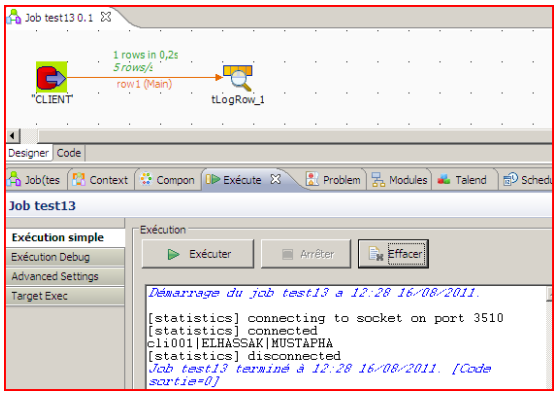



- Ensuite on modifie une à une les autres propriétés du composant en utilisant les variables de contexte déjà créées.




Voilà, notre job est de nouveau prêt à l'emploi, il suffit de

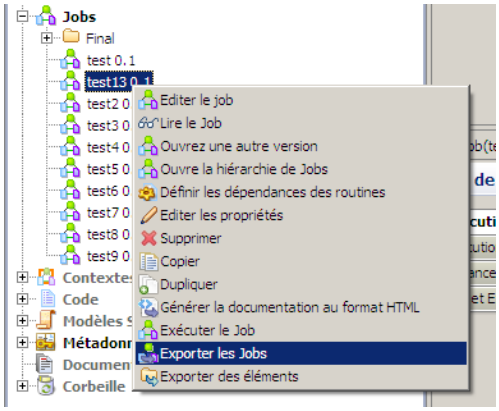
l'exécuter.



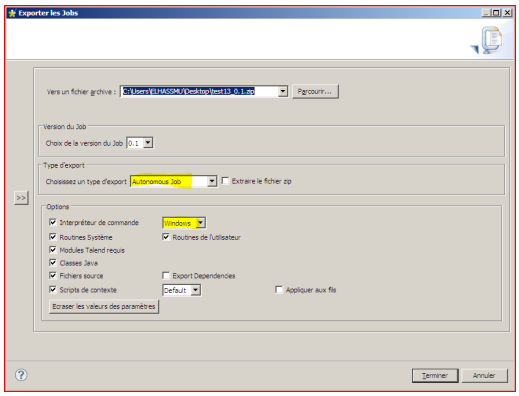

## 3.4. Export du job en batch

Cette opération est assez simple à réaliser avec Talend Open Studio.

- En effet on sauvegarde notre job puis on le ferme, ensuite on fait dessus un clic droit, puis on clique sur "Exporter les jobs".

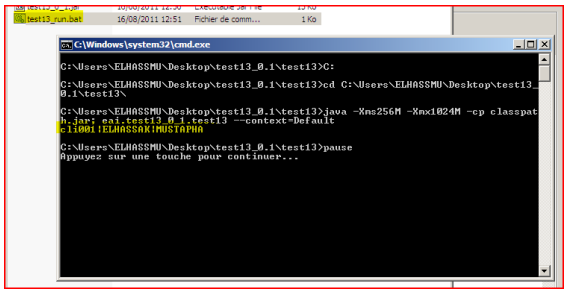



- La fenêtre d'export s'affiche, on tape le chemin du fichier exporté, on choisit le type d'export "Autonomous Job" et on spécifie la plate-forme "Windows" avant de cliquer sur "Terminer".

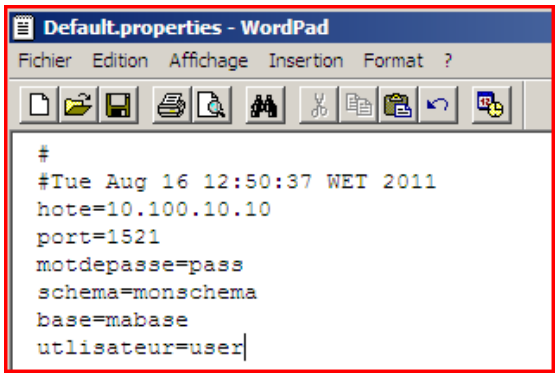



Notre job est maintenant exporté et on peut aller voir les fichiers générés sur notre système.

Parmi les fichiers générés on trouve un fichier **.bat** qui permet d'exécuter le job.




Revenons à nos variables de contexte, certains diront : "à quoi peuvent-elles bien servir ?" Et bien c'est ici qu'on verra toute leur puissance. En regardant dans les fichiers générés on peut retrouver un dossier "contexts" avec dedans un fichier **Default.properties** si on s'amuse à ouvrir ce fichier voilà ce qu'on y voit :




En effet on y retrouve toutes nos variables de contexte déclarées, et il suffit de modifier ce fichier pour modifier le comportement de notre job, si par exemple la base de données a changé d'adresse alors pas besoin de rouvrir le job avec Talend Open Studio, de le modifier puis de le régénérer. Non, il suffit juste de modifier l'entrée "hote" dans le fichier **Default.properties** et le tour est joué.

## 4. Conclusion

Merci d'avoir lu tout ce document :), j'espère qu'il a pu aider quelques-uns de ses lecteurs, au moins pour les débutants sur TOS. Ceci dit pour les gens qui souhaiteraient apprendre encore plus je leur conseille d'aller sur le site officiel de Talend où il y'a plusieurs autres tutos très intéressants, sans oublier le forum sur Developpez où vous pouvez poser toutes vos questions !

*Retrouvez l'article de Mustapha El Hassak en ligne : Lien 114*

# Liens

Lien 01 : http://bruno-orsier.developpez.com/tutoriels/TDD/pentaminos
Lien 02 : http://maven.apache.org/download.html#Installation
Lien 03 : http://thierry-leriche-dessirier.developpez.com/tutoriels/java/methode-3t/
Lien 04 : http://thierry-leriche-dessirier.developpez.com/tutoriels/java/3t-en-pratique/
Lien 05 : http://www.developpez.com/actu/32636/
Lien 06 : http://www.developpez.com/actu/37341/
Lien 07 : http://www.developpez.com/actu/36818/
Lien 08 : http://www.developpez.com/actu/36240/
Lien 09 : http://www.developpez.com/actu/37293/
Lien 10 : http://www.developpez.net/forums/d1105498/systemes/autres-systemes/mobiles/microsoft-negocierait-redevance-15-smartphone-sous-android-samsung/
Lien 11 : http://www.developpez.net/forums/d1120343/club-professionnels-informatique/actualites/google-rachete-motorola-mobility/
Lien 12 : http://developer.android.com/reference/android/media/SoundPool.html
Lien 13 : http://developer.android.com/reference/android/media/MediaPlayer.html
Lien 14 : ftp://ftp-developpez.com/jodul/tutoriels/android/creer-soundboard/fichiers/up.wav
Lien 15 : ftp://ftp-developpez.com/jodul/tutoriels/android/creer-soundboard/fichiers/coin.wav
Lien 16 : http://developer.android.com/reference/android/media/MediaPlayer.html#create%28android.content.Context,%20int%29
Lien 17 : http://jodul.developpez.com/tutoriels/android/creer-soundboard/
Lien 18 : http://dsilvera.developpez.com/tutoriels/android/creer-listview-avec-checkbox-et-gerer-evenements/
Lien 19 : http://xtext.itemis.com/xtext/language=en/36553/downloads
Lien 20 : http://www.eclipse.org/Xtext/documentation/1_0_0/xtext.html#DSL
Lien 21 : http://gkemayo.developpez.com/eclipse/intro-xtext/
Lien 22 : http://tcuvelier.developpez.com/tutoriels/php/symfony2/doctrine2/data-fixtures/
Lien 23 : http://jplu.developpez.com/tutoriels/web-semantique/introduction/
Lien 24 : http://web-semantique.developpez.com/tutoriels/
Lien 25 : http://dbpedia.org/
Lien 26 : http://www4.wiwiss.fu-berlin.de/dblp/
Lien 27 : http://opendatacommons.org/
Lien 28 : http://wiki.dbpedia.org/Downloads36
Lien 29 : http://api.sindice.com/v2/search?qt=term&q=Cat
Lien 30 : http://www.w3.org/TR/2008/NOTE-cooluris-20081203/
Lien 31 : http://sindice.com/main/submit
Lien 32 : http://void.rkbexplorer.com/submit/
Lien 33 : http://kwijibo.talis.com/voiD/submit
Lien 34 : http://pingthesemanticweb.com/
Lien 35 : http://tcuvelier.developpez.com/tutoriels/web-semantique/indexation-donnees-void/
Lien 36 : http://microformats.org/wiki/adr
Lien 37 : http://microformats.org/wiki/geo
Lien 38 : http://microformats.org/wiki/hatom
Lien 39 : http://microformats.org/wiki/haudio
Lien 40 : http://microformats.org/wiki/hcalendar
Lien 41 : http://microformats.org/wiki/hcard
Lien 42 : http://microformats.org/wiki/hlisting
Lien 43 : http://microformats.org/wiki/hmedia
Lien 44 : http://microformats.org/wiki/hnews
Lien 45 : http://microformats.org/wiki/hproduct
Lien 46 : http://microformats.org/wiki/hrecipe
Lien 47 : http://microformats.org/wiki/hresume
Lien 48 : http://microformats.org/wiki/hreview
Lien 49 : http://microformats.org/wiki/rel
Lien 50 : http://microformats.org/wiki/rel-license
Lien 51 : http://microformats.org/wiki/rel-nofollow
Lien 52 : http://microformats.org/wiki/rel-tag
Lien 53 : http://microformats.org/wiki/rel-directory
Lien 54 : http://microformats.org/wiki/rel-enclosure
Lien 55 : http://microformats.org/wiki/rel-home
Lien 56 : http://microformats.org/wiki/rel-payment
Lien 57 : http://microformats.org/wiki/robots-exclusion
Lien 58 : http://microformats.org/wiki/vote-links
Lien 59 : http://gmpg.org/xfn/
Lien 60 : http://microformats.org/wiki/xfolk
Lien 61 : http://gmpg.org/xmdp/
Lien 62 : http://microformats.org/wiki/xoxo
Lien 63 : http://microformats.org/code/hcalendar/creator
Lien 64 : http://microformats.org/code/hcard/creator
Lien 65 : http://hresume.weblogswork.com/hresumecreator/
Lien 66 : http://microformats.org/code/hreview/creator
Lien 67 : http://gmpg.org/xfn/creator
Lien 68 : http://web-semantique.developpez.com/tutoriels/microformats/guide/
Lien 69 : http://heureuxoli.developpez.com/office/word/vba-all/
Lien 70 : http://heureuxoli.developpez.com/office/xml/
Lien 71 : http://msdn.microsoft.com/en-us/library/aa752043%28v=VS.85%29.aspx
Lien 72 : http://support.microsoft.com/kb/269238
Lien 73 : http://msdn.microsoft.com/en-us/library/aa741317.aspx
Lien 74 : http://msdn.microsoft.com/en-us/library/aa384106%28v=VS.85%29.aspx
Lien 75 : http://msdn.microsoft.com/en-us/library/aa384276%28v=vs.85%29.aspx
Lien 76 : http://silverlight.net/

Lien 77 : http://argyronet.developpez.com/office/vba/convention/
Lien 78 : http://a-pellegrini.developpez.com/tutoriels/coding-style/
Lien 79 : http://msdn.microsoft.com/en-us/library/aa752043%28v=VS.85%29.aspx
Lien 80 : http://j-willette.developpez.com/tutoriels/html/les-bases-du-html/
Lien 81 : http://arkham46.developpez.com/articles/office/officeweb/
Lien 82 : http://loufab.developpez.com/tutoriels/access/optimisation/
Lien 83 : http://loufab.developpez.com/tutoriels/access/operateur-in/
Lien 84 : http://www.blog.loicrebours.fr/index.php/2011/08/17/wp7-mango-samuser-avec-les-contacts/
Lien 85 : http://loicrebours.developpez.com/Users/Loïc/Desktop/2011102001mangotasks.odt/fichiers/MangoTasks.rar
Lien 86 : http://loicrebours.developpez.com/tutoriels/dotnet/windows-phone-7-mango-decouvrez-nouvelles-tasks/
Lien 87 : http://cpp.developpez.com/faq/cpp/?page=heritage#heritage_lsp
Lien 88 : http://cpp.developpez.com/faq/cpp/?page=heritage#HERITAGE_NVI
Lien 89 : http://loic-joly.developpez.com/articles/heritage-multiple/
Lien 90 : http://qt-project.org/
Lien 91 : http://www.developpez.net/forums/d1144709/c-cpp/bibliotheques/qt/vivats-qt-project/
Lien 92 : http://developer.qt.nokia.com/
Lien 93 : http://developer.qt.nokia.com/wiki/QtWhitepaper
Lien 94 : http://qt.nokia.com/
Lien 95 : http://qt-devnet.developpez.com/tutoriels/qt/livre-blanc/#architecture
Lien 96 : http://qt.nokia.com/qt-in-use
Lien 97 : http://doc.qt.nokia.com/latest/gallery.html
Lien 98 : http://qt.developpez.com/doc/latest/exemples.html
Lien 99 : http://qt.developpez.com/doc/latest/layout.html
Lien 100 : http://qt.developpez.com/doc/latest/object.html
Lien 101 : http://qt.developpez.com/doc/latest/signalsandslots.html
Lien 102 : http://qt.developpez.com/doc/latest/qt4-mainwindow.html
Lien 103 : http://qt.developpez.com/doc/latest/designer-manual.html
Lien 104 : http://qt.nokia.com/products/developer-tools
Lien 105 : http://qt.developpez.com/doc/latest/qtdesigner.html
Lien 106 : http://qt.developpez.com/doc/latest/exemples/
Lien 107 : http://qt-devnet.developpez.com/tutoriels/qt/livre-blanc/
Lien 108 : http://www.4d.com/fr/blog/4dv123-available.html
Lien 109 : http://www.4d.com/fr/downloads/products.html
Lien 110 : http://www.developpez.net/forums/d1139468/environnements-developpement/autres-edi/4d/4d-v12-3-disponible/
Lien 111 : http://www.4d.com/fr/company/events/confdevfragenda.html
Lien 112 : http://www.developpez.net/forums/d1139481/environnements-developpement/autres-edi/4d/conference-developpeur-paris/
Lien 113 : http://haskouse.developpez.com/tutoriels/etl/talend-open-studio/creation-job/
Lien 114 : http://haskouse.developpez.com/tutoriels/etl/talend-open-studio/utilisation-contexte/