Developpez

Le Mag

Édition de décembre - janvier 2011/2012.
Numéro 37.
Magazine en ligne gratuit.

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

Sommaire

Éditorial

Pour bien démarrer cette nouvelle année, le magazine refait le plein de nos meilleurs articles, critiques de livres, questions/réponses, news. Ils sont à découvrir ou redécouvrir.

Profitez-en bien !

La rédaction

Article MS Office

Comment positionner un formulaire à un endroit déterminé

Ce texte s'adresse à des utilisateurs Access débutants qui veulent s'initier à la programmation.

par **Claude Leloup**
Page 18

Article DotNet

Le débogage sous Visual Basic 6 & Visual Basic pour Application (1re partie)

Tout ce que vous devez savoir sur le débogage et la gestion des erreurs sous Visual Basic 6.

par **DarkVader**
Page 52

Introduction à Tycho : construction automatique d'un product Eclipse

Ce tutoriel est une introduction à Tycho, un plugin pour Maven permettant de construire des bundles OSGi et des plugins Eclipse. Nous montrons par l'exemple comment construire automatiquement des exécutables Eclipse (appelés « product ») par l'intermédiaire de Tycho.

1. Introduction

Historiquement la construction automatique de bundles OSGi ou de plugins Eclipse passait par l'utilisation d'un ensemble de tâches ANT fournies par la plateforme Eclipse. Malheureusement la manipulation de ces tâches ANT n'était pas simple et l'utilisation dans un outil d'intégration continue (Jenkins, Cruise Control par exemple) était rendue difficile. À côté de cela, Maven l'outil de construction de binaires a démontré qu'il était aisé de compiler, de jouer des tests et de gérer les dépendances dans l'univers Java. Toutefois l'utilisation de Maven dans la plateforme Eclipse était impossible du fait qu'Eclipse se base sur une architecture OSGi. Avec l'arrivée de Tycho, qui est un plugin Maven, il est désormais possible d'utiliser toute la puissance de Maven pour construire des bundles OSGi et des plugins Eclipse.

Dans ce tutoriel, je vous propose l'utilisation du plugin Tycho dans le cas de la construction d'exécutables Eclipse RCP également appelés product. Nous montrons par l'exemple toutes les étapes nécessaires en insistant sur la structuration du projet en différents plugins et comment générer des binaires selon une version spécifique de la plateforme Eclipse.

Ce tutoriel suppose que vous disposiez des connaissances de base sur les technologies suivantes :

- développement de plugins avec la plateforme Eclipse (SWT/JFace, Eclipse RCP...), voir la page des tutoriels Eclipse de Developpez.com pour une mise à niveau : Lien 01 ;
- construction de binaires avec Maven.

2. Installation des outils

Cette section présente tous les outils utilisés dans le cadre de ce tutoriel. Nous signalons volontairement pour chaque outil, la version utilisée lors de la réalisation de ce tutoriel.

- **Eclipse RCP / Plugin** : l'environnement de développement Java pour développer des plugins (version 3.7.0).
- **Maven 3** : l'outil de construction automatique de binaires.
- **Tycho** : le plugin pour Maven pour construire des binaires de bundles OSGi et de plugins Eclipse.
- **M2Eclipse**: l'intégration de Maven pour Eclipse.

Nous détaillons pour chacun de ces outils la procédure d'installation et de configuration.

2.1. Eclipse RCP / Plugin

L'environnement de développement Eclipse est naturellement utilisé puisqu'il intègre la plateforme de développement de plugins. Par ailleurs, nous utiliserons la version Indigo qui est la version courante au moment de l'écriture de ce tutoriel.

Le téléchargement de l'environnement de développement est obtenu sur le site de la fondation Eclipse : Lien 02.

Pour l'installation, décompresser l'archive dans le répertoire où vous installez généralement vos applications (par exemple : c:\program files).

2.2. Maven 3

Maven est un outil de construction de binaires pour la plateforme de développement Java. Il se distingue des autres outils dans le sens où il utilise une approche déclarative. En effet, le contenu et la structure d'un projet Java sont décrits.

Pour utiliser Tycho, vous devez obligatoirement installer la version de Maven 3 disponible en téléchargement sur le site de la fondatation Apache : Lien 03.

Pour l'installation, suivre les étapes élémentaires suivantes :

- décompresser l'archive dans le répertoire où vous installez généralement vos applications (par exemple : c:\program files) ;
- créer une variable d'environnement **M2_HOME** qui pointe sur le répertoire de Maven (par exemple : c:\program files\maven3) ;
- ajouter le répertoire binaire de Maven à la variable d'environnement **PATH.**

2.3. Tycho

Tycho est un plugin pour Maven pour construire des binaires de bundles OSGi et de plugins Eclipse. Comme indiqué précédemment il ne peut fonctionner qu'à partir de la version 3 de Maven.

Pour l'installation, il suffit de compléter le fichier *pom.xml* au niveau de la balise <plugins>. Nous étudierons ce point plus précisément dans la suite au niveau de la section 3.

```
<build>
 ...
 <plugins>
 <plugin>
```

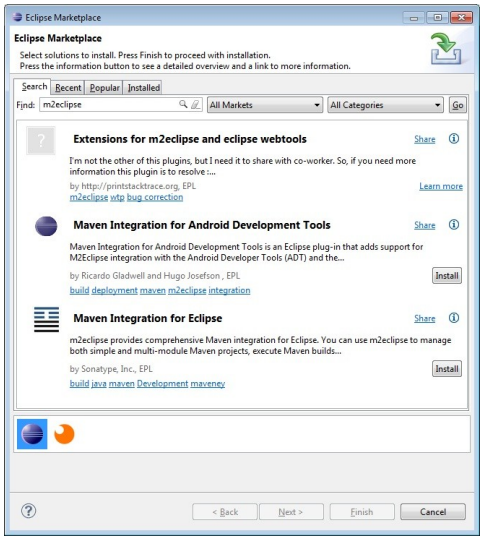
```
  <!-- enable tycho build extension -->
  <groupId>org.sonatype.tycho</groupId>
  <artifactId>tycho-maven-plugin</artifactId>
  <version>${tycho-version}</version>
  <extensions>true</extensions>
 </plugin>
 </plugins>
 </build>
</project>
```

2.4. M2Eclipse

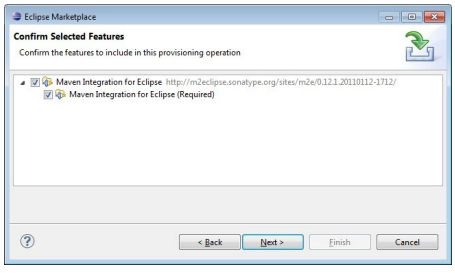
M2Eclipse est une intégration de Maven pour l'environnement de développement Eclipse. Son utilisation permet d'employer des commandes Maven directement dans Eclipse et permet également de rendre la manipulation du fichier pom.xml plus aisée puisqu'une interface graphique sous forme de formulaire remplace la représentation XML assez verbeuse.

Pour l'installation, nous utiliserons l'outil Marketplace de l'éditeur Eclipse. Voici les étapes.

- Ouvrir l'outil Marketplace via le menu *Help -> Eclipse Marketplace...*
- Depuis la zone de texte *Find*, saisir la valeur *M2Eclipse*, puis cliquer sur "rechercher". Vous devriez obtenir le résultat présenté sur la capture d'écran ci-dessous :



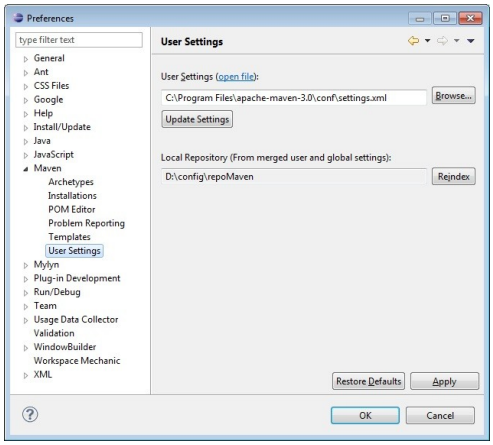
- Cliquer sur *Install* depuis le plugin *Maven Integration for Eclipse.*
- Après un certain temps de recherche, vous devriez obtenir le résultat suivant, cliquer sur *Next.*



- Accepter la licence d'autorisation, puis cliquer sur *Finish*. Le plugin va être installé et il vous sera demandé de redémarrer Eclipse, ne pas accepter le redémarrage et quitter Eclipse.
- Avant de relancer Eclipse, modifier le fichier *eclipse.ini* situé à la racine de votre répertoire d'installation d'Eclipse afin de préciser le répertoire de votre JDK. Le plugin M2Eclipse utilise ce chemin pour exécuter ses commandes.

```
-vm
C:/Program Files
(x86)/Java/jdk1.6.0_18/bin/javaw.exe
```

- Redémarrer Eclipse, ouvrir le menu des préférences (*Window -> Preferences*) et sélectionner les options liées à Maven.
- Depuis les options Maven, afficher les éléments liés à *User Settings* et préciser dans la zone *User Settings* votre fichier *settings.xml* (lié au répertoire d'installation de Maven 3) et cliquer sur *Update Settings*.



3. Construction des plugins

Dans la suite, nous présentons l'utilisation de Tycho au travers d'une application RCP contenant une vue affichant un arbre. L'objectif final est de générer automatiquement une archive contenant notre product Eclipse.

Nous insistons dans un premier temps sur la décomposition de l'application en plusieurs projets, chacun contenant un fichier de description *pom.xml*. Cette décomposition est présentée ci-dessous :

- *keulkeul.tychorcpdemo.aggregator* : regroupe tous les autres projets ;
- *keulkeul.tychorcpdemo.rcp* : contient le plugin RCP ;
- *keulkeul.tychorcpdemo.parent* : le projet Parent au niveau Maven ;
- *keulkeul.tychorcpdemo.feature* : décrit un projet feature ;
- *keulkeul.tychorcpdemo.repository* : contient la description d'un product et l'update site.

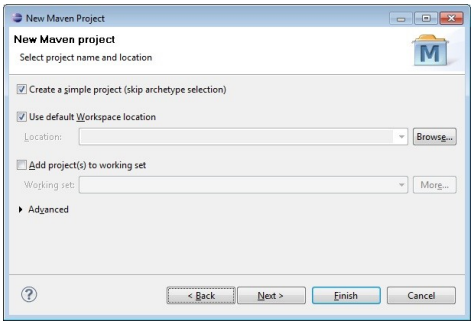
Dans un second temps, nous détaillons le contenu de chaque fichier de description *pom.xml* afin d'indiquer les paramétrages pour Tycho.

3.1. Création d'un projet Agrégateur (keulkeul.tychorcpdemo.aggregator)

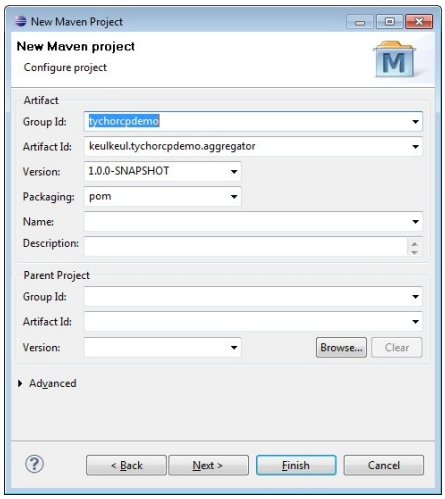
Ce projet Agrégateur aura comme seul but de contenir tous les sous-projets (plugins, feature, tests, repository p2...). Du point de vue Maven il s'agira de décrire dans le fichier *pom.xml* tous les sous-modules .

Veuillez suivre la démarche ci-dessous.

- Ouvrir l'assistant de création de projet File -> New -> Other...
- Choisir l'élément Maven Project de la catégorie Maven, l'écran ci-dessous doit apparaître :



- Cocher l'option *Create a simple project* (*skip archetype selection*), puis cliquer sur *Next.*



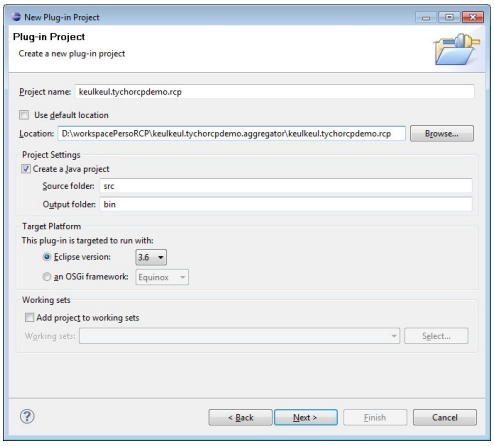
- Dans le champ de texte *Group Id* saisir la valeur *tychorcpdemo.*
- Dans le champ de texte *Artifact Id* saisir la valeur *keulkeul.tychorcpdemo.aggregator.*
- Au niveau de la sélection *Packaging* sélectionner la valeur *pom* puis cliquer sur *Finish.*

Un nouveau projet Maven sera créé contenant un fichier *pom.xml*. Nous modifierons ce fichier à chaque fois que nous ajouterons un nouveau module.

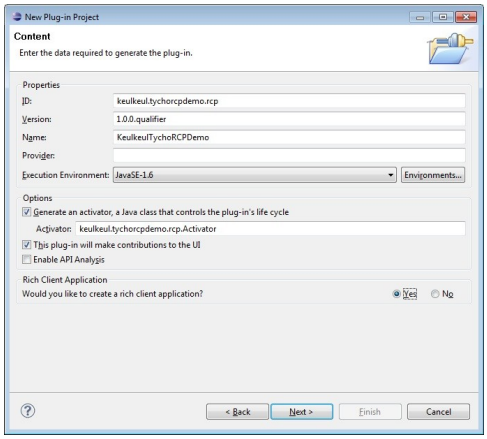
3.2. Création du plugin RCP (keulkeul.tychorcpdemo.rcp)

Ce plugin est particulier dans le sens où il s'agit d'un RCP (Rich Client Application) ce qui signifie qu'il peut être utilisé seul. Comme l'idée n'est pas d'apprendre à créer un plugin RCP nous utiliserons un exemple patron.

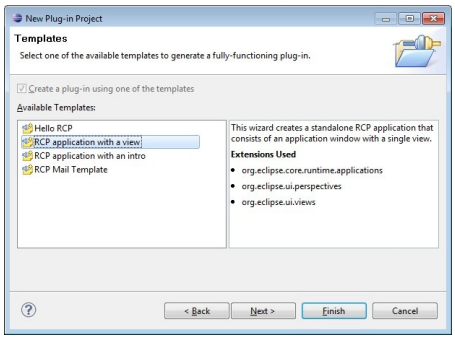
- Ouvrir l'assistant de création de projet File -> New -> Other...
- Choisir l'élément Plug-in Project depuis la catégorie Plug-in Development, puis cliquer sur *Next*. L'écran ci-dessous doit apparaître :



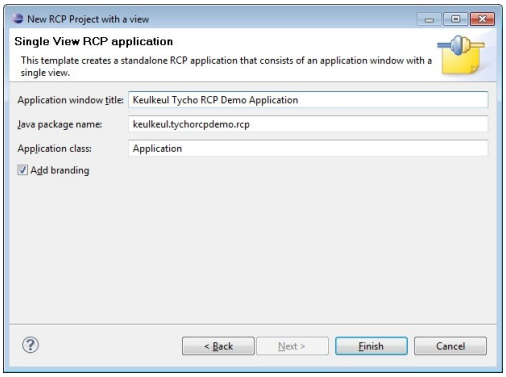
- Dans le champ *Project Name* saisir la valeur *keulkeul.tychorcpdemo.rcp.*
- Décocher l'option *Use default location.*
- Créer un répertoire *keulkeul.tychorcpdemo.rcp* à la racine du répertoire du projet agrégateur créé précédemment, puis cliquer sur *Next*. L'écran suivant doit apparaître :



- Cocher la case *Would you like to create a rich client application ?* puis cliquer sur *Next*. L'écran suivant doit apparaître :



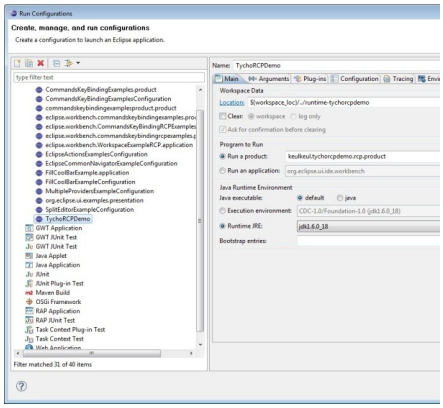
- Choisir comme patron *RCP application with a view* puis cliquer sur *Next*. L'écran suivant doit apparaître :



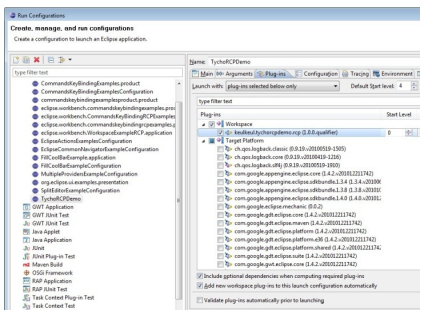
- Dans le champ *Application window title* saisir la valeur *Keulkeul Tycho RCP Demo Application.*
- Sélectionner l'option *Add branding* ce qui permettra d'être identifié comme un *product* lors de la création de la configuration d'exécution, puis cliquer sur *Finish*.

Un nouveau projet *keulkeul.tychorcpdemo.rcp* vient d'être créé. Pour s'assurer que ce plugin RCP fonctionne nous allons créer une configuration d'exécution.

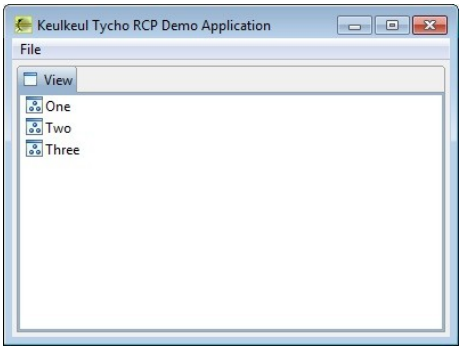
- Ouvrir l'écran de création des configurations. Depuis le menu principal *Run -> Run Configurations* ... L'écran suivant doit apparaître :



- Créer une nouvelle configuration de type *Eclipse Application* dont le nom sera *TychoRCPDemo.*
- Choisir depuis *Run a product* la valeur *keulkeul.tychorcpdemo.rcp.product* (uniquement disponible si vous aviez coché l'option *Add branding*. Sélectionner l'onglet *Plug-ins* et vous devriez obtenir l'écran suivant :



- Choisir depuis Launch with la valeur plug-ins selected below only.
- Décocher tous les plugins présents depuis le nœud racine Target Platform.
- Sélectionner uniquement le plugin *keulkeul.tychorcpdemo.rcp* depuis le nœud racine Workspace.
- Cliquer sur *Add Required Plug-ins* pour ajouter uniquement les plugins requis à l'exécution de ce plugin RCP.
- Cliquer sur *Run*. Vous devriez obtenir l'écran suivant :



- Ajouter un nouveau fichier Maven pom.xml à la racine de ce projet (*keulkeul.tychorcpdemo.rcp*) dont le contenu est le suivant :

```
<project>
 <modelversion>4.0.0</modelversion>
 <parent>
 <groupid>tychorcpdemo</groupid>
 <artifactid>keulkeul.tychorcpdemo.parent</artifa
ctid>
 <version>1.0.0-SNAPSHOT</version>
 <relativepath>../keulkeul.tychorcpdemo.parent/po
m.xml</relativepath>
 </parent>
 <groupid>tychorcpdemo</groupid>
 <artifactid>keulkeul.tychorcpdemo.rcp</artifacti
d>
 <version>1.0.0-SNAPSHOT</version>
 <packaging>eclipse-plugin</packaging>
</project>
```

- Comme vous pouvez le constater, ce fichier *pom.xml* fait référence à un *pom.xml* parent que nous définirons plus tard. Intéressons-nous d'abord à l'ajout d'un projet *keulkeul.tychorcpdemo.rcp* comme module au projet *keulkeul.tychorcpdemo.aggregator.* Par conséquent modifier le fichier pom.xml du projet *keulkeul.tychorcpdemo.aggregator* de cette manière :

```
<project>
 <modelVersion>4.0.0</modelVersion>
 <groupId>tychorcpdemo</groupId>
 <artifactId>keulkeul.tychorcpdemo.aggregator</ar
tifactId>
 <version>1.0.0-SNAPSHOT</version>
 <packaging>pom</packaging>
 <modules>
 <module>keulkeul.tychorcpdemo.parent</module>
 <module>keulkeul.tychorcpdemo.rcp</module>
```

```
 </modules>
</project>
```

- De même nous profitons de cette modification pour ajouter un autre module *keulkeul.tychorcpdemo.parent.*

3.3. Création d'un projet parent (keulkeul.tychorcpdemo.parent)

Le projet parent a pour objectif de contenir toutes les configurations propres à Tycho dont tous les plugins auront besoin. Il contiendra également les liens vers les entrepôts p2.

La démarche de création de ce projet est identique à celle utilisée pour créer le projet *keulkeul.tychorcpdemo.aggregator* à la différence que ce projet doit être placé à la racine du projet agrégateur. Pour les valeurs à donner dans le nouveau fichier pom.xml, suivre les indications données ci-dessous.

- Dans le champ de texte Group Id saisir la valeur *tychorcpdemo.*
- Dans le champ de texte *Artifact Id* saisir la valeur *keulkeul.tychorcpdemo.parent.*
- Au niveau de la sélection *Packaging* sélectionnez la valeur *pom* puis cliquer sur *Finish.*
- Compléter le fichier *pom.xml* généré par les informations liées à la configuration de Tycho

```
<project>
 <modelVersion>4.0.0</modelVersion>
 <groupId>tychorcpdemo</groupId>
 <artifactId>keulkeul.tychorcpdemo.parent</artifa
ctId>
 <version>1.0.0-SNAPSHOT</version>
 <packaging>pom</packaging>

 <properties>
 <tycho-version>0.11.0</tycho-version>
 </properties>

 <repositories>
 <!-- configure p2 repository to resolve against
-->
 <repository>
  <id>helios</id>
   <layout>p2</layout>
   <url>http://download.eclipse.org/releases/heli
os/</url>
 </repository>
 </repositories>
 <build>
 <plugins>
 <plugin>
  <!-- enable tycho build extension -->
  <groupId>org.sonatype.tycho</groupId>
  <artifactId>tycho-maven-plugin</artifactId>
  <version>${tycho-version}</version>
  <extensions>true</extensions>
 </plugin>
 </plugins>
 </build>
</project>
```

À ce niveau vous devriez obtenir la structure de fichiers suivante :

```
keulkeul.tychorcpdemo.aggregator \-
 pom.xml
 keulkeul.tychorcpdemo.parent \-
 pom.xml
 keulkeul.tychorcpdemo.rcp \-
 src \-
 pom.xml
```

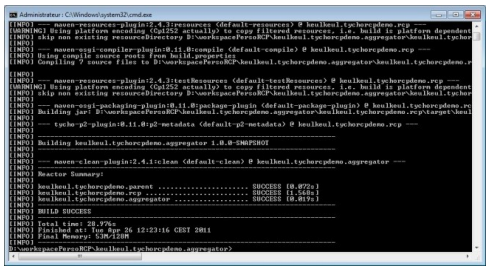
3.4. Démarrer une construction du projet via Maven

Nous allons volontairement construire le projet en utilisant l'invite de commandes de Windows au lieu d'utiliser le plugin M2Eclipse. En effet, Tycho est surtout utilisé pour construire le projet final via un outil d'intégration continue. C'est une façon de s'assurer que cette construction fonctionne en dehors de l'environnement Eclipse.

Ouvrir l'invite de commandes de Windows et se placer à la racine du répertoire parent. Saisir la ligne de commande ci-dessous :

```
mvn clean install
```

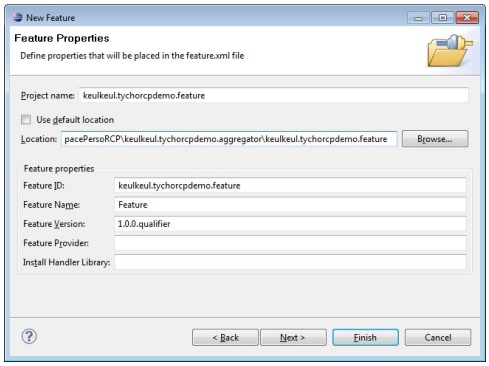
Normalement si tout se passe bien vous devriez obtenir le message suivant :



3.5. Création d'un projet feature (keulkeul.tychorcpdemo.feature)

Ce projet de type feature va nous permettre de regrouper dans une feature l'ensemble des plugins de notre application. Il faut admettre que pour l'instant il n'y en a pas énormément. L'idée est que si vous souhaitez ajouter de nouveaux plugins, vous n'aurez qu'à modifier cette feature.

- Ouvrir l'assistant de création de projet File -> New -> Other...
- Choisir l'élément *Feature Project* depuis la catégorie Plug-in Development, puis cliquer sur *Next*. L'écran ci-dessous doit apparaître :



- Choisir comme nom de projet *keulkeul.tychorcpdemo.feature.*

- Personnaliser le répertoire de travail pour être à la racine du projet *keulkeul.tychorcpdemo.aggregator*.
- Choisir depuis la liste des plugins disponibles (Initialize from the plug-ins list) le plugin *keulkeul.tychorcpdemo.rcp* puis cliquer sur *Finish*.

Un nouveau projet *keulkeul.tychorcpdemo.feature* vient d'être créé.

- Ajouter un nouveau fichier Maven pom.xml à la racine de ce projet dont le contenu est le suivant :

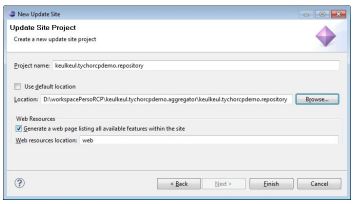
```
<project>
 <modelVersion>4.0.0</modelVersion>
 <parent>
 <groupId>tychorcpdemo</groupId>
 <artifactId>keulkeul.tychorcpdemo.parent</artifa
ctId>
 <version>1.0.0-SNAPSHOT</version>
 <relativePath>../keulkeul.tychorcpdemo.parent/po
m.xml</relativePath>
 </parent>
 <groupId>tychorcpdemo</groupId>
 <artifactId>keulkeul.tychorcpdemo.feature</artif
actId>
 <version>1.0.0-SNAPSHOT</version>
 <packaging>eclipse-feature</packaging>
</project>
```

- Ajouter ce nouveau projet comme module en complétant le pom.xml du projet *keulkeul.tychorcpdemo.aggregator*.
- Vérifier que l'application se construit correctement via un mvn clean install.

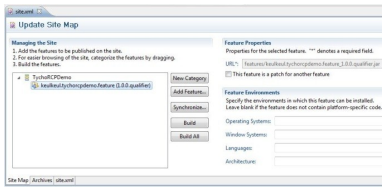
3.6. Création d'un projet update site (keulkeul.tychorcpdemo.repository)

Ce nouveau projet a pour objectif de créer un update site de notre projet. Cela permettra ainsi d'utiliser l'outil de mise à jour pour installer nos nouveaux plugins.

- Ouvrir l'assistant de création de projet File -> New -> Other...
- Choisir l'élément Plug-in Project depuis la catégorie Plug-in Development, puis cliquer sur *Next*. L'écran ci-dessous doit apparaître :



- Choisir comme nom de projet *keulkeul.tychorcpdemo.repository*.
- Personnaliser le répertoire de travail pour être à la racine du projet *keulkeul.tychorcpdemo.aggregator*.
- Cocher l'option *Generate a web page listing all available features within the site* puis cliquer sur *Finish*.
- Depuis le nouveau projet généré, renommer le fichier site.xml en category.xml.
- Ouvrir le fichier site.xml et depuis l'onglet Site Map créer une nouvelle catégorie appelée TychoRCPDemo.
- Depuis cette nouvelle catégorie, ajouter la feature *keulkeul.tychorcpdemo.feature* créée précédemment (voir capture d'écran ci-dessous).



- Ajouter un nouveau fichier Maven pom.xml à la racine de ce projet dont le contenu est le suivant :

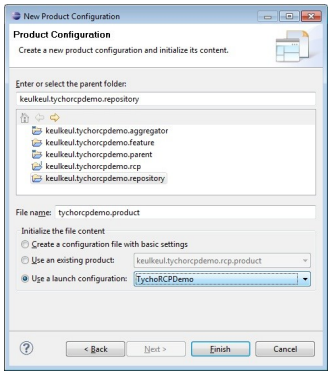
```
<project>
 <modelVersion>4.0.0</modelVersion>
 <parent>
 <groupId>tychorcpdemo</groupId>
 <artifactId>keulkeul.tychorcpdemo.parent</artifa
ctId>
 <version>1.0.0-SNAPSHOT</version>
 <relativePath>../keulkeul.tychorcpdemo.parent/po
m.xml</relativePath>
 </parent>
 <groupId>tychorcpdemo</groupId>
 <artifactId>fr.ensma.lisi.tychorcpdemo.repositor
y</artifactId>
 <version>1.0.0-SNAPSHOT</version>
 <packaging>eclipse-repository</packaging>
</project>
```

- Ajouter ce nouveau projet comme module en complétant le pom.xml du projet *keulkeul.tychorcpdemo.aggregator*.
- Vérifier que l'application se construit correctement via un *mvn clean install*.

3.7. Création d'un fichier product (keulkeul.tychorcpdemo.repository)

Le fichier product contient toutes les informations relatives à la construction d'un exécutable Eclipse (plateformes supportées, le splashscreen, le nom de l'application, les images...).

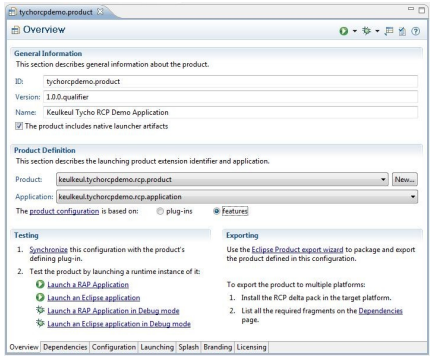
- Ouvrir l'assistant de création de projet File -> New -> Other...
- Choisir l'élément *Product Configuration* depuis la catégorie Plug-in Development, puis cliquer sur *Next*. L'écran ci-dessous doit apparaître :



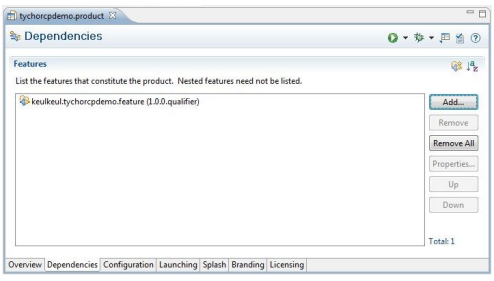
- Choisir le projet *keulkeul.tychorcpdemo.repository.*
- Saisir la valeur tychorcpdemo.product dans le champ File name.
- Choisir pour l'option *Use a launch configuration* la valeur TychoRCPDemo puis cliquer sur *Finish*.
- Éditer en mode texte le fichier tychorcpdemo.product (sans l'éditeur de configuration de product) et ajouter les instructions suivantes qui permettront de démarrer les plugins adéquats au lancement de l'application RCP

```
<configurations>
 <plugin id="org.eclipse.core.runtime"
autoStart="true" startLevel="0" />
 <plugin id="org.eclipse.equinox.common"
autoStart="true" startLevel="2" />
 <plugin id="org.eclipse.osgi" autoStart="true"
startLevel="-1" />
</configurations>
```

- Depuis l'onglet Overview du mode édition du fichier tychorcpdemo.product, saisir la valeur TychoRCPDemo.product.
- Modifier l'option The Product configuration is based on par la valeur features.
- S'assurer que la valeur *keulkeul.tychorcpdemo.rcp.product* est sélectionnée pour l'option Product et que la valeur *keulkeul.tychorcpdemo.rcp.application* est sélectionnée pour l'option Application. Voir capture d'écran ci-dessous :



- Depuis l'onglet Dependencies choisir la valeur *keulkeul.tychorcpdemo.feature*. Voir capture d'écran ci-dessous :



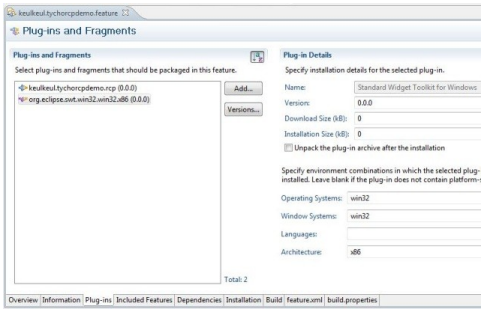
- Pour construire un product Eclipse selon un environnement donné (OS/WS/Arch) compléter le fichier pom.xml du projet *keulkeul.tychorcpdemo.repository* comme présenté ci-dessous :

```
<build>
 <plugins>
  <plugin>
   <groupId>org.sonatype.tycho</groupId>
   <artifactId>tycho-p2-director-
plugin</artifactId>
   <version>${tycho-version}</version>
   <executions>
    <execution>
     <id>materialize-products</id>
     <goals>
      <goal>materialize-products</goal>
     </goals>
    </execution>
   </executions>
  </plugin>
 </plugins>
 </build>
</project>
```

Finalement depuis le projet feature (*keulkeul.tychorcpdemo.feature*) il nous reste à ajouter le fragment SWT correspondant à la plateforme Windows (notre plateforme de tests) de façon à construire le product pour cette plateforme.

À noter que pour le support de différentes versions de Windows (64 bits par exemple) ou le support de différents systèmes d'exploitation (Linux, MAC OS X...), il sera nécessaire de réitérer cette étape.

- Ouvrir le fichier feature.xml et afficher l'onglet Plug-ins.
- Ajouter la feature org.eclipse.swt.win32.win32.x86 et définir la valeur win32 dans le champ Operating Systems, la valeur win32 dans le champ Window Systems et la valeur x86 dans le champ Architecture. Voir capture d'écran ci-dessous pour visualiser le résultat attendu :



- Vérifier que l'application se construit correctement via un mvn clean install. Depuis le répertoire *keulkeul.tychorcpdemo.repository/target/products/TychoRCPDemo.product/win32/win32/x86* vous devriez obtenir le résultat de cette construction. Un product Eclipse construit automatiquement à l'aide de Maven/Tycho.

4. Conclusion

Ce tutoriel a présenté une introduction à l'utilisation du plugin Tycho pour construire automatiquement un product Eclipse. Nous avons insisté sur la structuration des différents plugins qui composent notre projet et sur le

contenu XML à placer dans les différents fichiers de description spécifiques à Maven.

De nombreuses fonctionnalités restent à découvrir concernant Tycho et notamment la possibilité d'exécuter des tests unitaires, de gérer des dépendances de bibliothèques tierces, de générer un product Eclipse selon la version de la plateforme, de déployer les nouveaux plugins dans un update site, et de faire référence à des plugins depuis un update site donné.

5. Liens

Vous trouverez ci-dessous en ensemble de liens qui peuvent compléter ce tutoriel :

- Apprendre à construire un plugin : Lien 04.
- Quelques billets sur Tycho : Lien 05.
- Site de Tycho : Lien 06.

6. Sources

Les sources de ce tutoriel sont disponibles ici : FTP (Lien 07) ou HTTP (Lien 08).

Retrouvez l'article de Mickael BARON en ligne : Lien 09

Android devient numéro 1 en France

Plus d'un smartphone vendu sur deux sous l'OS de Google dans le monde

Le succès d'Android se confirme en France. Le système d'exploitation de Google est devenu l'OS mobile numéro 1 sur le territoire français, selon un récent rapport publié par le cabinet d'étude des médias Médiamétrie.

En six mois, les achats de nouveaux smartphones sous Android par les consommateurs ont été deux fois plus nombreux qu'au premier trimestre 2011, permettant ainsi à l'OS de devancer pour la première fois iOS en France avec 40 % de parts de marché.

L'attrait des terminaux mobiles sous l'OS de Google au cours de ces derniers mois se confirme aussi de façon globale à travers le monde entier.

Les résultats d'une nouvelle étude du cabinet d'analyse Gartner, au cours du troisième trimestre de cette année, révèlent qu'Android détient une part de marché de 52,5 %, soit le double des ventes de smartphones Android au troisième trimestre 2010.

Pour Gartner, cette croissance d'Android se justifie par la diversité des terminaux mobiles sous l'OS, un affaiblissement de l'environnement concurrentiel, le manque de nouveaux produits pour les systèmes d'exploitation alternatifs comme Windows Phone 7 et BlackBerry OS.

La seconde place est occupée par Symbian, qui enregistre une chute de près de 20 points en un an, pour se retrouver à 16,9 % de parts de marché au cours de cette période.

On constate également une baisse des ventes de l'iPhone d'Apple, dont la part représente désormais 15 % pour le troisième trimestre, contre 16,6 % pour la même période de l'année dernière. Cette baisse s'explique selon l'institut Gartner par l'attente du nouvel iPhone par les consommateurs.

RIM, le constructeur des téléphones BlackBerry, recule aussi de quatre points sur un an avec 11 % de parts de marché pour le troisième trimestre. Gartner constate néanmoins que les ventes des téléphones de la firme sont restées stables entre les deux trimestres.

Worldwide Smartphone Sales to End Users by Operating System in 3Q11 (Thousands of Units)

Operating System	3Q11 Units	3Q11 Market Share (%)	3Q10 Units	3Q10 Market Share (%)
Android	60,490.4	52.5	20,544.0	25.3
Symbian	19,500.1	16.9	29,480.1	36.3
iOS	17,295.3	15.0	13,484.4	16.6
Research In Motion	12,701.1	11.0	12,508.3	15.4
Bada	2,478.5	2.2	920.6	1.1
Microsoft	1,701.9	1.5	2,203.9	2.7
Others	1,018.1	0.9	1,991.3	2.5
Total	**115,185.4**	**100**	**81,132.6**	**100**

Globalement, le marché des téléphones mobiles enregistre une hausse de 5,6 % par rapport à la même période de l'année dernière, avec près de 440,5 millions de dispositifs vendus. Les ventes de smartphones, quant à elles, ont quasiment explosé avec une hausse de 42 % par rapport à 2010. 115 millions de smartphones ont été vendus au troisième trimestre.

Côté constructeurs, Samsung devient le premier constructeur de smartphones pour cette période. Nokia demeure néanmoins le premier fabricant de téléphones mobiles.

Commentez la news d'Idelways en ligne : Lien 10

Google publie le code source d'Android 4

Le kit de développement natif de l'OS supporte désormais les API de cette version

Sur la mailing-list d'Android, un ingénieur de Google annonce la publication progressive du code source d'Ice Cream Sandwish (Lien 11) sur son serveur Git public, d'où il peut être librement téléchargé.

Cette ouverture constitue un pas significatif pour Android, car il s'agit de la première fois qu'une base de code de l'OS, adaptée aux tablettes, soit accessible.
Android 4 représente un environnement unifié pour les smartphones et les tablettes. Pour ces dernières, une version d'Android a été dédiée : Honeycomb (Android 3) (Lien 12), mais son code source n'a jamais été publié (Lien 13). Google avait sélectionné les partenaires qui ont eu droit de l'utiliser, à l'instar de Motorola pour sa tablette Xoom.

Ces sources désormais disponibles incluent tout l'historique de l'évolution du code. Il est donc possible de remonter aux sources de Honeycomb que Google n'a pas tagué afin de dissuader les développeurs de l'utiliser. L'ingénieur Jean-Baptiste Queru explique dans cette annonce que cette version « *est quelque peu incomplète. Nous voulons que chacun se concentre sur Ice Cream Sandwish* », déclare-t-il.

Lors de la sortie des premiers appareils sous Honeycomb, beaucoup d'observateurs avaient fustigé le modèle open source très contrôlé. Google publie cette fois le code source avant le lancement commercial du Galaxy Nexus, mais n'améliore pas la gouvernance du projet, toujours à sens unique.

Parallèlement, le Kit de Développement Natif de l'OS vient d'avoir une nouvelle révision qui expose aux applications C et C++ les nouvelles API introduites par Android 4.0 : Lien 14.

Les instructions pour télécharger les sources d'Android 4.0.1 sont consultables sur cette page : Lien 15

Télécharger la nouvelle révision du NDK : Lien 16

Commentez la news d'Idelways en ligne : Lien 17

Les dernier tutoriels et articles

Android et les Services Web SOAP

Cet article va vous présenter comment appeler des Services Web SOAP au sein de votre application Android et comment parser le résultat de l'appel.

1. Préambule

L'appel de Services Web SOAP peut s'avérer très utile au sein d'une application Android. Ici nous allons voir comment appeler un Service Web permettant de connaître la météo du jour. Avant cela, une petite mise au point est nécessaire.

Tout d'abord, qu'est-ce qu'un Service Web ?

Un Service Web est une fonctionnalité exécutable à distance pouvant être développée dans divers langages comme Java, .NET... Ces services peuvent être appelés à partir d'un langage différent de celui du service. Par exemple, il est possible d'appeler un service codé en Java au sein d'un code en PHP. Les fonctionnalités de ces Web_Services peuvent être des interactions avec une base de données, fournir des informations au programme appelant le service. Un Service Web SOAP nécessite une WSDL contrairement à des services REST.

Qu'est-ce qu'une WSDL?

Le WSDL est une description fondée sur le XML qui indique comment utiliser le service.
Le WSDL sert à décrire :
- le protocole de communication (SOAP RPC ou SOAP orienté message) ;
- le format de messages requis pour communiquer avec ce service ;
- la définition des méthodes qu'il est possible d'appeler ;
- la localisation du service.

Une description WSDL est un document XML qui commence par la balise "definitions" et qui contient les balises suivantes :
- "binding" : définit le protocole à utiliser pour invoquer le service web ;
- "port" : spécifie l'emplacement actuel du service ;
- "service" : décrit un ensemble de points finals du réseau.

Des outils sont-ils nécessaires?

Pour appeler un Service Web SOAP, une bibliothèque est nécessaire vu qu'il n'existe pas un tel outil dans le SDK d'Android. Il s'agit de la bibliothèque kSoap2 disponible à l'adresse suivante : kSoap2 (Lien 18).
Elle permettra également de parser la réponse du service.

2. Appel d'un service

Nous allons supposer qu'une WSDL (Web Services Description Language, ou langage de description de services) est disponible à l'adresse http://mon-exemple-web-services/wsdl?WSDL, celle-ci proposant la méthode getMeteo(String ville). Voyons comment appeler ce service :

```
  public class AppelService {

  private static final String NAMESPACE =
"http://mon-site-web.fr";
  private static final String URL = "http://mon-
exemple-web-services/wsdl.WSDL";
  private static final String SOAP_ACTION =
"getMeteo";
  private static final String METHOD_NAME =
"getMeteo";

  private String getMeteo(String ville) {

    try {
      SoapObject request = new
SoapObject(NAMESPACE, METHOD_NAME);
      request.addProperty("ville", ville);

      SoapSerializationEnvelope envelope = new
SoapSerializationEnvelope(SoapEnvelope.VER11);
      envelope.setOutputSoapObject(request);

      AndroidHttpTransport androidHttpTransport =
new AndroidHttpTransport(URL);
      androidHttpTransport.call(SOAP_ACTION,
envelope);

    } catch (Exception e) {
      Log.e("getMeteo", "", e);
    }
  }
}
```

Rien de spécial à retenir dans ce code à part qu'il utilise intégralement la bibliothèque. Afin de pouvoir l'utiliser, il vous faudra juste remplacer les constantes par vos URL et noms de méthodes, adaptées à votre WSDL et si nécessaire ajouter des paramètres à votre requête.

Voilà, nous savons donc maintenant comment appeler un Service Web

3. Parser une réponse SOAP

Il est fort intéressant de savoir appeler un Service Web, encore faut-il savoir comment parser la réponse. Ici, nous voulons donc parser la réponse SOAP se présentant de la manière suivante :

```
<meteo>
        <temps>beau</temps>
</meteo>
```

```
  public class AppelService {

  private static final String NAMESPACE =
"http://mon-site-web.fr";
  private static final String URL = "http://mon-
exemple-web-services/wsdl.WSDL";
  private static final String SOAP_ACTION =
"getMeteo";
  private static final String METHOD_NAME =
"getMeteo";

  private String getMeteo(String ville) {
    String meteo = null;
    try {
      SoapObject request = new
SoapObject(NAMESPACE, METHOD_NAME);
      request.addProperty("ville", ville);

      SoapSerializationEnvelope envelope = new
SoapSerializationEnvelope(SoapEnvelope.VER11);
      envelope.setOutputSoapObject(request);

      AndroidHttpTransport androidHttpTransport =
new AndroidHttpTransport(URL);
      androidHttpTransport.call(SOAP_ACTION,
envelope);
      SoapObject objetSOAP =
(SoapObject)envelope.getResponse();
      meteo = this.parserObjet(objetSOAP);

    } catch (Exception e) {
      Log.e("getMeteo", "", e);
    }
  }

  private String parserObjet(SoapObject objet) {
    SoapObject meteoObjet =
(SoapObject)positionSoap.getProperty("meteo");
    String meteo =
meteoObjet.getProperty("temps").toString();

    return meteo;
  }

}
```

Un simple appel à la méthode getProperty permet de récupérer un autre objet SoapObject ou directement la valeur d'une propriété.

4. Conclusion

Vous avez pu voir qu'il est très simple de faire appel à des Services Web SOAP et de parser les résultats. Ici, il s'agit d'un exemple simple mais même si vous faites face à un plus compliqué, vous ne serez pas perdu ! Dans un prochain article, nous verrons comment appeler des Services REST.

Retrouvez l'article de Michel Dirix en ligne : Lien 19

Introduction à l'écosystème Java

Cet article fait un vaste tour d'horizon du monde java et des technologies qui y sont liées afin de permettre aux nouveaux arrivants de se faire une idée d'ensemble.

1. Introduction

La très grande richesse de Java constitue également l'un de ses défauts du point de vue de la personne qui débute et qui ne sait pas trop par où commencer. De fait, les tutoriels pour apprendre les bases sont nombreux (Lien 20), mais lorsqu'il faut aller au-delà, chaque cours va être lié à un aspect spécifique de Java, et il n'est pas toujours aisé de voir les liens entre toutes les facettes, en particulier lorsque l'on débute. Le but de ce tutoriel est donc de présenter une approche parmi d'autres de la découverte progressive de l'écosystème Java et également du J2EE. La complexité de ce tutoriel n'est pas linéaire, elle augmente à mesure que des choses plus compliquées sont abordées. Arrêtez-vous lorsque vous commencerez à ne plus comprendre, et n'hésitez pas à cliquer sur les liens qui renvoient vers des tutoriels ayant trait aux sujets abordés ;).

2. Ma première classe

Ce paragraphe est davantage destiné à situer le point de départ de la progression du tutoriel qu'à fournir un point de départ dans l'apprentissage de Java. De fait, dans l'ensemble de ce tutoriel, on supposera que les bases de la programmation orientée objet (POO) en Java sont connues du lecteur.

Pour l'écriture de la première classe Java, point n'est besoin d'utiliser un outillage complexe. À partir du moment où vous avez installé un JDK et configuré les variables d'environnement (Lien 21), de simples éditeurs de texte tels que Notepad sous Windows ou Gedit sous Ubuntu suffisent pour écrire le premier programme.

```
public class PremiereClasse {
    public static void main(String[] args) {
        System.out.println("Bonjour ceci est mon
premier essai avec Java");
    }
}
```

Après sauvegarde dans un fichier que l'on nomme donc "PremiereClasse.java" on peut compiler puis exécuter en ligne de commande à l'aide de :

```
$ javac PremiereClasse.java
$ java PremiereClasse
Bonjour ceci est mon premier essai avec Java
```

On peut coder un moment de cette manière. Tant que l'on ne va utiliser que les bibliothèques fournies avec le JDK, ça ne pose aucun souci. Il suffit de placer toutes ses classes à la suite dans le même fichier (une seule sera publique, celle qui contient la méthode main). Cependant vous éprouverez rapidement le besoin d'utiliser un éditeur de texte qui fournisse au minimum la coloration syntaxique afin de rendre la lecture de code plus aisée. Gedit sous Linux le fait déjà, mais il existe des solutions plus poussées comme Geany (qui est multiplateforme) ou Notepad++ (pour Windows et Mac).

3. Utilisation d'un environnement de développement

Si vous progressez en Java, vous allez rapidement vous retrouver confronté aux limites de l'approche précédente et commencer à utiliser un environnement de développement. Il s'agit d'un logiciel qui va automatiser tout ce que vous faisiez à la main en ligne de commande et vous fournir des outils d'aide au développement qui vont grandement vous faciliter la vie. Le plus connu est Eclipse (Lien 22), mais il existe plusieurs autres possibilités (Lien 23).

Dans un environnement de développement (EDI, ou en anglais : IDE), vous allez pouvoir très facilement compiler et tester vos programmes, utiliser des fichiers différents pour chaque classe, structurer les classes en packages et sous-packages , exporter l'ensemble du projet sous forme d'un jar exécutable, ajouter des bibliothèques tierces, et dans le cas d'un projet J2EE, déployer le projet dans un serveur d'applications, toutes opérations qui étaient déjà possibles en ligne de commande, mais qui vont se trouver en pratique grandement facilitées par l'utilisation d'un EDI. Dans la suite de ce tutoriel nous nous concentrerons sur Eclipse, qui est le plus répandu (mais pas forcément le plus simple). Vous trouverez ici de nombreux cours pour apprendre à débuter avec Eclipse : Lien 24.

Quelques raccourcis clavier pour vous mettre l'eau à la bouche :

- CTRL+ESPACE = autocomplétion. Vous commencez à écrire un mot et Eclipse le termine pour vous ;
- ALT+MAJ+J = Javadoc. Permet de générer automatiquement le squelette des commentaires qui ont trait au code survolé par le curseur ;
- CTRL+MAJ+O = Import automatique des dépendances requises par votre classe pour peu qu'elles soient contenues dans des bibliothèques situées dans le CLASSPATH du projet.

Le Classpath est une variable d'environnement du projet qui recense l'ensemble des chemins dans lesquels le compilateur va pouvoir trouver des classes ou des bibliothèques du projet.

Attention à l'UTF-8. Par défaut, Eclipse enregistre les

fichiers dans le codage de caractères système. C'est-à-dire en gros ISO pour Windows et UTF-8 pour les autres. C'est pourquoi il est conseillé de choisir un codage donné et de s'y tenir si vous ne souhaitez pas avoir de problèmes avec des caractères accentués qui s'affichent mal (je vous recommande le tout UTF-8).

4. Premières interfaces graphiques

En général, on fait ses armes sur AWT et (surtout) SWING, deux frameworks graphiques inclus dans le JDK, qui permettent la création relativement rapide d'interfaces graphiques. Pour quelqu'un qui débute, le niveau de difficulté peut sembler augmenter, car il y a un certain nombre de nouveaux concepts à assimiler, notamment ceux de Layout (comment organiser les contrôles de l'interface) (Lien 25) et d'EventListener (ou comment relier un bouton à une méthode) (Lien 26). Si vous venez d'un langage assez unifié comme le .Net, vous allez commencer à prendre conscience qu'en Java comme dans beaucoup d'autres langages, on code à partir de multiples briques fonctionnelles forgées par d'autres, et qui ne sont pas toujours faciles à manipuler. Si jamais vous avez la sensation de vous battre avec un framework, dites-vous que ça peut valoir le coup de chercher à mieux comprendre quand, par qui, comment et pourquoi il a été conçu ;). Pour vous aider à démarrer, il y a pléthore de tutoriels sur les interfaces graphiques Java : Lien 27.

À noter qu'alternativement à SWING, vous pouvez aussi vous tourner vers SWT (Lien 28), qui a été développé par IBM pour pallier certains inconvénients de SWING. On peut également mentionner ici les RCP (RCP signifie "Rich CLient Platform"), qui fournissent des interfaces standard et fournissent un important package logiciel prenant en compte les concepts d'extensibilité, de docking, d'onglets, de menus, de préférences, etc. mais se situent un niveau au-dessus de SWING et SWT en termes d'abstraction. Citons par exemple les plateformes Eclipse RCP (Lien 29) et Netbeans RCP, sur lesquelles sont basés les IDE du même nom. Pour aller plus loin : Lien 30.

5. Traduction au sein de mon application (i18n et l10n)

Envie de créer un programme qui soit utilisable par des gens parlant différents langages ? Il est relativement aisé de parvenir à ce but. L'astuce est toute simple : là où d'ordinaire vous écriviez en dur dans le code la phrase à afficher, vous allez en lieu et place utiliser une clé de traduction et utiliser la classe ResourceBundle (Lien 31) pour faire le lien avec la traduction correspondante que vous aurez placée dans un fichier de propriétés situé dans le classpath. Ceci est davantage développé dans le chapitre 24. L'internationalisation (Lien 32) du cours Développons en Java de J.M.Doudoux : Lien 33. Pour aller plus loin, il est également intéressant de mentionner ce tutoriel : Lien 34.

Comme le mentionne la documentation de la classe PropertyResourceBundle (Lien 35), il est préférable que les fichiers de propriétés soient encodés au format ISO-8859-1

Si vous vous demandez ce que veulent dire i18n et l10n, sachez que ces sigles barbares sont des raccourcis pour InternationalizatioN et LocalisatioN. À chaque fois on a gardé la première et la dernière lettre, et entre les deux on a mis le nombre de lettres intermédiaires.

6. Un peu d'architecture

Arrivés à un certain stade de développement, avec un projet sans cesse croissant en taille et en complexité, vous allez éprouver le besoin de découper le projet en plusieurs couches fonctionnelles et d'organiser vos classes au sein de ces couches afin de mieux dégager la structure d'ensemble et rendre la compréhension générale plus aisée. Le mécanisme des packages est d'une grande aide pour cela. Une architecture très courante est l'architecture dite "trois tiers" : elle consiste à construire une application en trois couches : la couche dite "de présentation" gère l'interface graphique, la couche dite "métier" contient le cœur de l'application, et la couche dite "d'accès aux données" permet de lire/persister les données nécessaires à l'application (typiquement l'accès à une base de données).

Ayez à l'esprit que pour réussir à créer une application dotée d'une bonne architecture, il faut y réfléchir avant de commencer à coder. C'est toute la phase de modélisation, préliminaire nécessaire à la création de toute application un tant soit peu complexe. Pour cette raison, il y a de nombreux tutoriels sur la question : Lien 36. Mais attention, il n'est pas toujours bon de se lancer dans une modélisation très complexe, cela doit vraiment dépendre de vos besoins. Abandonnez l'idée du design parfait (qui n'existe pas), car vous aurez forcément besoin de retoucher des détails pendant que vous coderez. Mieux vaut définir une architecture générale et spécifier en détail le fonctionnement des algorithmes les plus complexes, mais inutile d'aller plus loin.

Vous entendrez probablement parler aussi de Design Patterns : Lien 37. Il s'agit tout simplement des problèmes courants que peut rencontrer un développeur et des moyens reconnus par la communauté comme les plus pertinents pour y répondre. De mon expérience, la plupart ne sont pas immédiatement accessibles au débutant. Il est nécessaire d'avoir un minimum d'expérience et avoir vraiment été confronté au problème pour saisir la pertinence d'un pattern donné.

7. Bibliothèques externes et frameworks

Java est un langage très populaire et relativement ancien, il existe donc de nombreux bibliothèques et frameworks qui implémentent les besoins les plus courants du développeur et lui permettent de se concentrer sur la partie métier, c'est-à-dire ce qui l'intéresse vraiment et qui a trait au but dans lequel il développe, et non pas des trucs bateaux comme les entrées/sorties, la connexion Internet, l'export sous forme d'images ou de PDF, l'écriture d'emails, etc. La différence que je verrais entre une bibliothèque et un framework est la suivante : une bibliothèque est une archive jar que vous allez rajouter dans votre classpath, et dont vous allez utiliser les classes directement. Exemple typique, jdbc (qui permet d'accéder à une base de données). Alors qu'un framework, pour prendre une analogie ça va être comme un programme déjà existant au cœur duquel vous insériez le vôtre. Le framework se chargeant de toutes les opérations communes et vous apportant des fonctionnalités intéressantes comme l'inversion de contrôle (ce n'est plus vous qui appelez les

classes du framework, mais le framework qui appelle vos classes) et l'injection de dépendances (utile en programmation par contrats : vous déclarez un objet d'après l'interface qu'il doit implémenter et vous utilisez son setter pour lui fournir son implémentation véritable), ce qui vous permettra un meilleur découplage des différentes couches qui constituent l'architecture de l'application. Les frameworks sont particulièrement utiles dans l'univers J2EE (construction d'applications Java tournant dans un serveur Web). Exemple typique : Spring (Lien 38).

Pour les plus curieux, quelques explications complémentaires sur l'inversion de contrôle. L'explication qui en a été donnée ici : "ce n'est plus vous qui appelez les classes du framework, mais le framework qui appelle vos classes" est réductrice car elle ne présente qu'une facette de l'inversion de contrôle. Voici une explication plus détaillée :
le principe consiste à briser une dépendance bidirectionnelle (ou cyclique), ou encore à briser une dépendance unidirectionnelle non voulue comme une couche métier qui invoquerait directement des méthodes de la couche de l'IHM. Il existe deux grosses techniques :

- par échange de messages (cf. middleware) ;
- par contrat (cf. interface).

Dans le premier cas, il s'agit d'avoir un pont de communication abstrait via lequel les éléments s'échangent des messages. Dans le second cas, on détermine un jeu d'interfaces (ou de classes abstraites) qui seront implémentées par la couche invoquée de manière directe. Ensuite cette même couche fournit à la couche qui en a besoin les implémentations à utiliser (souvent par injection).

8. Fichiers de configuration et annotations

Tôt ou tard, si vous continuez à programmer en Java, vous allez avoir besoin d'utiliser des fichiers de configuration et/ou des annotations. Le but est de décrire le comportement du programme et la manière dont il va interagir avec d'autres programmes. Typiquement vous pouvez utiliser un fichier de configuration qui vous est propre pour indiquer des chemins d'accès, ou des paramètres que vous voulez pouvoir changer sans avoir besoin de recompiler le programme. Mais la première fois que vous aurez besoin d'utiliser un fichier de configuration sera sans doute pour configurer un framework ou une bibliothèque.

On distingue généralement deux types de fichiers de configuration : les fichiers XML et les fichiers de propriétés. Les fichiers de propriétés sont les plus simples : ils contiennent un ensemble de paires CLÉ=VALEUR à raison d'une par ligne. Les développeurs utilisent souvent des noms de clé qui reflètent la catégorie de celle-ci. Par exemple "monappli.macouchemetier.coefficients.usure=15". De cette manière il est très aisé de savoir à quoi va servir la clé (et d'éviter que deux clés nommées usure ayant trait à deux choses différentes ne se télescopent). Les fichiers XML sont un peu plus complexes, mais il ne faut pas en avoir peur. La structure est similaire à celle d'un fichier HTML et elle est généralement plus simple. L'avantage du XML est qu'il permet d'organiser facilement les données de configuration. Ce qu'on lui reproche généralement est sa verbosité : de nombreuses balises sont nécessaires à la description d'une information même simple. Si vous vous lancez dans le J2EE, vous ne pourrez pas en faire l'économie, puisque vous aurez nécessairement besoin d'un fichier web.xml pour décrire le comportement de votre application au serveur d'applications dans lequel elle sera déployée.

Venons-en aux annotations : Lien 39. Elles sont apparues avec la cinquième version de Java vers fin 2004. Elles visent à réduire le volume des fichiers de configuration (voire à s'en passer totalement), en écrivant directement dans le code à l'aide d'une balise spéciale des informations liées à son comportement. Ce qu'il faut garder en tête, c'est que les annotations ne sont pas un remède miracle, et il ne faut pas verser dans le tout « annotations » comme certains font, mais savoir les utiliser avec discernement. Le danger est le risque d'éparpiller des informations de configuration un peu partout dans le programme alors qu'elles seraient plus claires en étant centralisées dans un unique fichier de configuration. C'est en particulier le cas des annotations qui décrivent le comportement des classes les unes vis-à-vis des autres. À l'inverse, il y a peu d'intérêt à reporter dans un fichier de configuration des annotations propres au fonctionnement interne d'une classe donnée.

9. Tests unitaires

Les tests unitaires sont une partie importante des applications que les développeurs ont trop souvent tendance à oublier par manque de temps : Lien 40. Il s'agit tout simplement d'une classe de test qui va vérifier que chaque méthode d'une classe effectue bien le travail demandé. C'est particulièrement utile en cas de refactoring pour se rendre compte immédiatement des problèmes qui ont pu arriver. Par ailleurs les tests unitaires contiennent de précieuses informations sur le comportement attendu de l'application qui sont plus utiles que de longs commentaires. Une bonne pratique consiste à écrire ses tests unitaires avant d'écrire les classes auxquelles ils correspondent. Cela permet tout à la fois :

- de se poser les bonnes questions : "Quel est le comportement attendu de ma classe ?" plutôt que de commencer par modéliser et d'adapter les méthodes a posteriori ;
- d'être sûr d'avoir son test unitaire fonctionnel une fois la classe finie, et de ne pas être influencé dans le sens "je fais un test qui correspond à ce que j'avais en tête quand j'ai fait la méthode", ce qui peut parfois constituer un piège, puisque l'on ne vérifie pas la conformité aux spécifications.

Cette manière de programmer est appelée le TDD (Test Driven Developpement).

En Java, le framework communément utilisé pour les tests unitaires est Junit (Lien 41), mais il y a également TestNG (Lien 42), qui est une variante intéressante. De nombreux plugins pour Eclipse rendent aisée l'utilisation des tests unitaires. Citons tout d'abord le plugin pour JUnit lui-même (généralement fourni par défaut avec Eclipse), mais également MoreUnit, qui facilite le lien entre une classe et son test unitaire et la génération de mocks, ainsi

qu'Infinitest, qui permet de relancer automatiquement le dernier test effectué à chaque enregistrement d'une modification de fichier.

Rapidement vous pourrez rencontrer le problème suivant : lorsqu'une classe dépend d'une autre classe, comment tester juste cette classe précise sans tester aussi la dépendance ? La solution passe par l'utilisation de stubs (qui simulent le comportement d'une classe donnée) ou de mocks (qui vont rejouer une série de comportements préétablis). Citons EasyMock (Lien 43), Jmockit (Lien 44), et Mockito (Lien 45) parmi les bibliothèques de génération de mocks répandues.

10. Traces logicielles (logging)

Lorsque l'on teste un programme, on a besoin de savoir ce qu'il se passe. Au début on utilise l'instruction "System.out.println()" pour écrire dans la console des informations sur le déroulement du programme, mais très rapidement, cela ne suffit plus car il est ennuyeux de devoir recompiler pour le désactiver et qu'il serait également souhaitable de pouvoir définir des niveaux de criticité des messages. C'est pourquoi une solution communément utilisée dans le monde de Java est la librairie commons-logging d'Apache (Lien 46). Elle contient toutes les classes que vous aurez besoin d'appeler depuis votre programme Java. Au lancement elle recherchera dans votre classpath une dépendance chargée de la réalisation effective de l'écriture ou l'affichage des traces. En général, on utilise log4j pour cette tâche : Lien 47. Il est également possible d'appeler log4j directement sans passer par commons-logging, mais généralement on s'en abstient pour éviter de rendre les traces du programme dépendantes d'une implémentation particulière.

Il existe d'autres solutions que commons-logging. On peut citer par exemple SLF4J qui est assez largement utilisé : Lien 48. Il permet de pallier certains défauts que l'on reproche parfois à commons-logging. De la même manière, il en existe aussi pour log4J, dont notamment le projet Logback (Lien 49), qui se pose comme le successeur non officiel de log4J.

11. Bases de données

La plupart des applications un tant soit peu évoluées vont avoir besoin de s'interfacer avec des bases de données. L'outil universel utilisé dans ce but est JDBC : Lien 50. Il est composé de deux parties distinctes : une API Java contenant les fonctions que vous aurez besoin d'appeler, et un driver propre à la base de données que vous allez utiliser. En général on commence à faire ses armes sur l'utilisation des bases de données en Java avec cet outil.

Cependant il faut savoir qu'il existe des API de plus haut niveau qui permettent de gérer plus aisément la persistance des objets métiers en base de données. Citons par exemple JPA (implémentation de la spécification de référence) (Lien 51), Hibernate (de loin la solution la plus répandue) (Lien 52) et IBatis (qui permet de contrôler finement les requêtes SQL exécutées) (Lien 53).

12. Outils de build

Maven (Lien 54) devient très vite incontournable dès lors que l'on commence à avoir un projet Java qui inclut de multiples bibliothèques. Il s'agit d'un outil qui permet d'inclure facilement les bibliothèques dont le projet à besoin, et de les configurer aisément. Il permet également de gérer les dépendances de manière transitive sans en oublier aucune, ce qui est un luxe que vous apprécierez lorsque vous vous serez suffisamment battu avec les "ClassNotFoundException", caractéristiques de l'oubli d'inclusion d'une bibliothèque.

Mais Maven ne s'arrête pas là. Il s'agit en effet d'un outil de build (en français: de construction) de projet hautement configurable, et qui peut s'interfacer à de nombreux outils d'analyse de code (comme Sonar : Lien 55) et d'intégration continue (comme Hudson (Lien 56) ou Cruise Control (Lien 57)), ce qui en fait un outil à connaître absolument pour tout développeur Java qui se respecte.

Alternativement à Maven, il convient de mentionner également ANT : Lien 58. Cet outil est plus ancien que Maven et reste d'autant plus répandu qu'il est aussi très utilisé pour construire des projets développés dans d'autres langages. Notons que les deux ne sont nullement incompatibles, il est possible de les adapter l'un à l'autre, par exemple pour migrer en douceur. Pour vous faire une idée des avantages/inconvénients des différents outils de build, voici un comparatif : Lien 59.

13. Java EE

Java EE (ou J2EE) désigne l'ensemble des technologies permettant de créer des applications d'entreprise. Le principe est simple : on veut éviter de devoir recoder dans chaque projet tout ce qui n'est pas spécifique au cœur de métier de l'application. Par conséquent, on va utiliser un serveur d'applications (qui va contenir tout le code non spécifique), et y embarquer notre application (centrée sur ce qui est propre à notre métier). Des fichiers de configuration permettent de décrire au serveur d'applications comment l'application doit se comporter.
J2EE sert souvent à la création d'applications Web, de manière assez similaire à ce que fait PHP. Elle est plus complexe à mettre en œuvre que PHP, mais elle apporte beaucoup plus de possibilités lorsque l'on recherche des fonctionnalités un peu poussées.

Il y a deux facettes dans J2EE. Celle qui est la plus connue est la facette "présentation Web". On l'aborde généralement en commençant par coder des servlets et des JSP suivant le modèle MVC : Lien 60. On va généralement utiliser un serveur d'applications dit "conteneur léger" qui n'implémente la spécification J2EE que dans ce domaine, comme Tomcat (Lien 61) ou Jetty.
L'étape suivante consiste à utiliser un framework tel que Spring (Lien 62) ou Seam (Lien 63) pour avoir la plupart des fonctionnalités requises par une application WEB trois tiers un petit peu plus complexe, et éventuellement un framework de présentation tel que JSF (obligatoire dans le cas de Seam, et en train de devenir un standard, JSP étant plus ou moins déprécié) (Lien 64) pour parfaire l'expérience utilisateur mais aussi celle du développeur.

Ce qui est moins connu, c'est que J2EE ne s'arrête pas là. La spécification touche aussi des aspects qui n'intéressent que les concepteurs des applications les plus complexes, comme la distribution d'une application sur plusieurs serveurs, avec des fonctionnalités comme le failover (en gros la tolérance aux pannes, si possible sans interruption du service), et le load-balancing (c'est-à-dire la répartition de charge, tant au niveau du trafic HTTP, que de la charge de l'application elle-même). Il est donc possible d'avoir une même application Java dont les différentes instances d'une même classe seront réparties sur différents serveurs. Pour faire tourner de telles applications, il est nécessaire d'utiliser un serveur d'applications dit "conteneur lourd" qui implémente la totalité de la spécification J2EE, tel que par exemple Glassfish et JBossAS. C'est dans ce contexte que vous pouvez être amené à manipuler des EJB (Lien 65) et JCA (Lien 66), qui lors de sa sortie était présentée par Sun comme "La solution" au problème d'intégration entre le monde J2EE et les systèmes d'information d'entreprise.

14. Conclusion

C'est la fin de ce tour d'horizon de l'écosystème Java, j'espère ne rien avoir oublié d'essentiel, mais normalement ça devrait suffire à quelqu'un débutant le Java pour se faire une idée de la complexité de ce monde et avoir un aperçu d'une manière de l'aborder progressivement.

Petite astuce pour finir, sachez qu'il existe de nombreux plugins Eclipse qui peuvent vous faciliter grandement la vie. Le plugin m2eclipse pour l'intégration de Maven, le plugin Instasearch pour effectuer des recherches dans les fichiers du projet et le plugin "properties editor" pour éditer des fichiers de propriétés avec un codage en UTF8 transparent, en sont de bons exemples.

Retrouvez l'article de Gauthier Perrineau en ligne : Lien 67.

Comment positionner un formulaire à un endroit déterminé

Ce texte s'adresse à des utilisateurs Access débutants qui veulent s'initier à la programmation.
Dans ce tutoriel nous allons apprendre :
- à positionner un formulaire à un endroit déterminé de l'écran ;
- à récupérer les coordonnées d'un formulaire affiché.

Nous examinerons en détail le fonctionnement de l'instruction DoCmd.MoveSize.
Nous aborderons les notions de twips et pixels.
Nous ferons appel à une API : GetWindowRect.
Le tout au travers d'un exemple concret : un formulaire construit pas à pas.

1. Avant-propos

Access n'offre pas spontanément beaucoup de moyens pour positionner un formulaire à un endroit déterminé de l'écran.

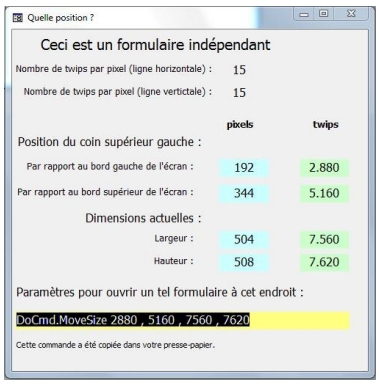
Si on veut y arriver, il faut faire appel à des API, c'est-à-dire des interfaces offertes par d'autres programmes (par exemple Windows) qui autorisent l'interaction avec Access. L'utilisation de ces techniques n'est pas facilement abordable par un débutant.

Voici les références de deux contributions sur Developpez.com qui abordent le sujet au moyen d'API :

- Argyronet a traité le sujet dans son tutoriel Comment positionner un formulaire ouvert depuis un autre, à droite de l'écran : Lien 68.
- Arkham46 dans la contribution suivante, donne tout le code pour positionner un formulaire sous un contrôle d'un autre formulaire : Lien 69.

Notre démarche consistera à utiliser une seule API et d'en expliquer le fonctionnement.

Comme exemple concret pour illustrer la théorie, nous allons construire pas à pas, ce formulaire gadget néanmoins utilisable :



Si on le déplace à l'écran ou si on le redimensionne, les données affichées sont instantanément actualisées.

Si vous voulez ouvrir un formulaire **indépendant** à un endroit donné avec des dimensions précises, vous le positionnez manuellement. Ensuite vous le recouvrez exactement avec cet utilitaire. La commande DoCmd.MoveSize avec les paramètres adéquats se trouve dans votre presse-papier.

2. Le cas banal

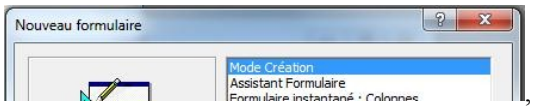
Pour illustrer notre propos, construisons un formulaire appelé ***fOuSuisJe***.

2.1. Construisons notre formulaire

Dans la fenêtre Access : onglet « Formulaires », clic sur 

,

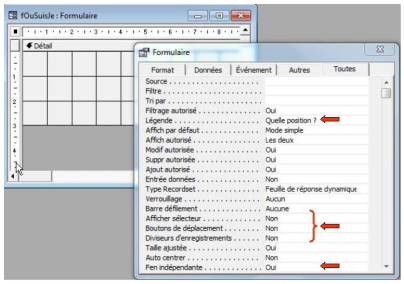
choisir Mode Création



clic sur  pour enregistrer sous le nom ***fOuSuisJe.***

Dans un premier temps, ce formulaire est totalement vide : il ne contient aucun contrôle et n'est rattaché à aucune source. Nous le garnirons au fur et à mesure des besoins de l'exposé.

Nous modifions quelques-unes des propriétés comme illustré ci-après :



Sauvegarder le formulaire et ouvrez-le à nouveau en double-cliquant sur son nom dans la fenêtre des objets.

Le formulaire s'affiche à l'écran à l'endroit où il se trouvait lorsque vous l'avez sauvegardé.

- Ouvrez le formulaire.
- Déplacez-le.
- Sauvegardez-le.
- Fermez-le.
- Ouvrez-le à nouveau.

Faites l'expérience

3. Positionner et dimensionner un formulaire à l'ouverture

3.1. La méthode MoveSize de l'objet Docmd

Cette méthode permet de déplacer ou de redimensionner la fenêtre du formulaire.

En d'autres mots, elle permet de positionner le formulaire à un endroit déterminé d'avance et de préciser la hauteur et la largeur de sa fenêtre.

Pour décrire, avec des mots, un rectangle affiché à l'écran, quatre données suffisent :

- la position horizontale du coin supérieur gauche ;
- la position verticale du coin supérieur gauche ;
- la largeur ;
- la hauteur.

Ce sont, précisément, les paramètres de la méthode *Docmd.MoveSize*.

Un exemple de syntaxe :

DoCmd.MoveSize 2835, 3405, 5670, 5100

Les unités de ces arguments sont exprimées en twips.

Twip est l'acronyme de TWentIeth of a Point soit 1/20^{e} de point. C'est une mesure d'affichage indépendante de la résolution de l'écran.
1 pouce (inch) = 1440 twips.
1 cm = ±567 twips.

Dans l'exemple, les paramètres expriment que l'on veut placer le coin supérieur gauche 5 cm à droite et 6 cm en dessous (nous expliquerons *par rapport à quoi,* plus bas) et que le formulaire aura 10 cm de large et 9 cm de hauteur.

3.2. Si le formulaire est une fenêtre indépendante

Reprenons notre formulaire ***fOuSuisJe*** et essayons de voir à quoi correspond une telle syntaxe.

Nous allons communiquer à Access, que l'événement « Sur ouverture » du formulaire ***fOuSuisJe*** doit déclencher l'instruction ***DoCmd.MoveSize 2835, 3405, 5670, 5100***.

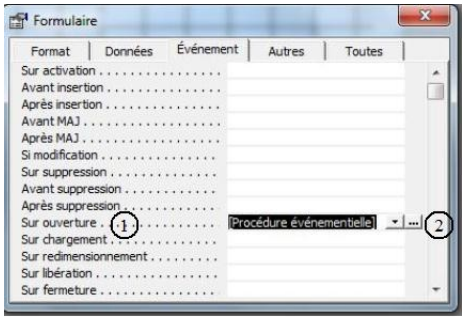
Nous demandons donc à Access que lorsqu'un utilisateur ouvre le formulaire ***fOuSuisJe***, ce dernier soit à l'endroit décrit et aux dimensions spécifiées dans les paramètres.

Nous allons coder cette instruction.

Ouvrons notre formulaire en mode construction.

Double-cliquons sur son coin supérieur gauche pour afficher les propriétés :

Une fenêtre s'ouvre et affiche les propriétés du formulaire. Cliquez sur l'onglet « Événement ».

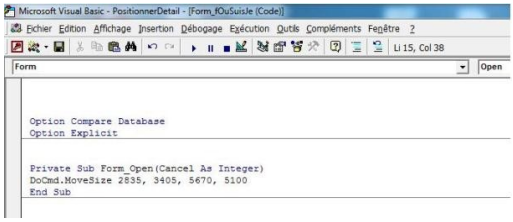


Double-cliquez sur le texte « Sur ouverture » (1) et cliquez ensuite sur les « … » (2) qui apparaissent à l'extrême droite de la ligne.

Vous ouvrez ainsi l'éditeur de code VBA, à l'endroit qui correspond au code associé à l'événement « Sur ouverture » (Form_Open) du formulaire.

Là où se situe le curseur, vous saisissez le code : ***DoCmd.MoveSize 2835, 3405, 5670, 5100.***

Vous obtenez finalement ceci :



Vous refermez la fenêtre de l'éditeur de code.

Vous enfoncez la touche <F5> pour passer en mode Formulaire. Ce dernier s'ouvre à 5 cm à droite et 6 cm en dessous du coin supérieur gauche de l'écran et sa fenêtre fait 10 cm de large et 9 cm de hauteur.

Si le formulaire est une fenêtre indépendante, le point de repère pour positionner avec ***DoCmd.MoveSize*** est le coin supérieur gauche de l'écran.

Ceci nous permet d'attirer l'attention sur l'importance de la propriété « Fen indépendante » pour déterminer la position qu'occupera à l'écran, le formulaire après exécution de l'instruction ***DoCmd.MoveSize***.

Pour constater où se situe le repère, nous allons modifier l'instruction :

DoCmd.MoveSize 0, 0, 5670, 5100

En français : déplacer le formulaire 0 twip à droite ; 0 twip en dessous ; avec une largeur de 10 cm ; avec une hauteur de 9 cm.

Pour accéder au code, enfoncez simultanément les touches <ALT> et <F11> (on note : ***<Alt-Key-F11>***).

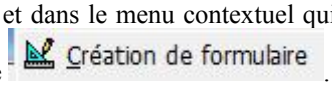
Modifiez l'instruction.

Fermez la fenêtre.

Ouvrez le formulaire, il s'ouvre maintenant dans le coin supérieur gauche de votre écran.

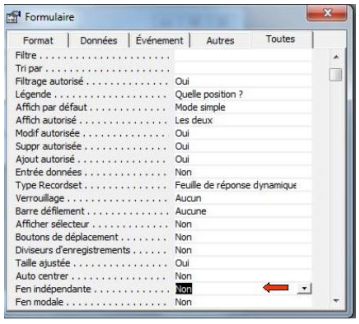
3.3. Et si le formulaire n'est pas indépendant ?

Essayons !

Basculons notre formulaire en mode création. Faites un clic droit sur le formulaire et dans le menu contextuel qui s'affiche, cliquez sur l'icône 

.

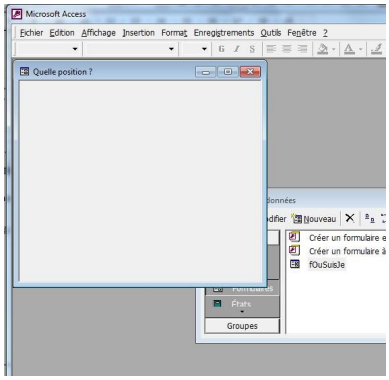
Affichez les propriétés et positionnez à « NON » Fen indépendante.



Fermez la fenêtre de l'éditeur de code. Fermez le formulaire en le sauvegardant.

Ouvrez le formulaire à nouveau.

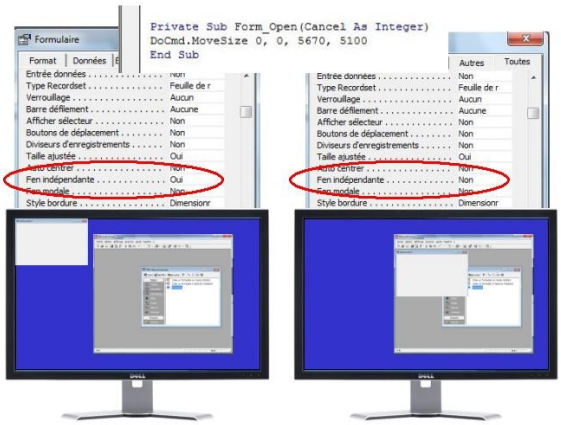
Cette fois la position du formulaire se situe à l'intérieur de la fenêtre de l'application ACCESS en dessous des barres d'outils et collé au bord gauche.



Le positionnement du formulaire avec la méthode MoveSize de l'objet Docmd dépend de la valeur attribuée à la propriété « Fen indépendante ».

Nous avons associé ***DoCmd.MoveSize*** à l'ouverture du formulaire, nous aurions pu l'associer à d'autres événements : le clic sur un bouton par exemple.

3.4. En résumé



4. Trouver la position et les dimensions d'un formulaire

Essayons de faire l'inverse. C'est-à-dire pouvoir exprimer où se trouve un formulaire ouvert et quelles sont sa largeur et sa hauteur.

C'est plus dur, car Access, du moins dans la version 2000 ne fournit pas, nativement, ce genre de renseignement.

Par contre, de telles informations sont disponibles dans le système d'exploitation Windows.

Windows API ou **WinAPI** est le nom donné par Microsoft à l'interface de programmation (API) qui permet à une application comme Access d'interagir avec le système d'exploitation Windows.

4.1. GetWindowRect : une API utile pour localiser un formulaire à l'écran

GetWindowRect

Cette fonction nous renseigne les coordonnés du coin supérieur gauche et du coin inférieur droit d'un rectangle affiché à l'écran.
Pour la déclarer dans notre application, il faut insérer ce code dans un module :

Déclaration de GetWindowRect

```
Declare Function GetWindowRect Lib "user32" Alias
"GetWindowRect" ( _
                        ByVal hwnd As Long, _
                        lpRect As RECT) As Long
```

Pour l'utiliser, il faut définir un type de données :

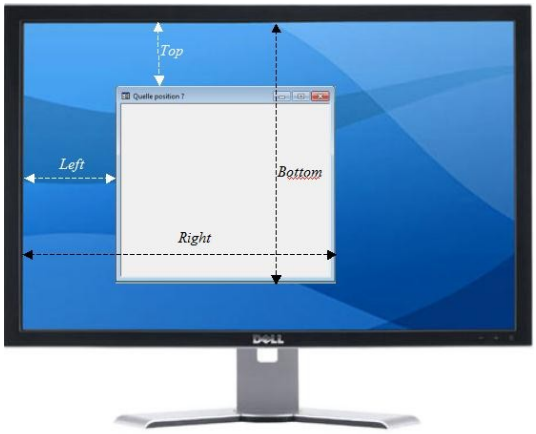
Type Rect
```
Type RECT
   Left As Long
   Top As Long
   Right As Long
   Bottom As Long
End Type
```

Il faudra l'appeler en donnant le paramètre hWnd du rectangle qui nous intéresse, en l'occurrence le rectangle occupé par notre formulaire.

Un hWnd est un nombre entier généré par Windows permettant d'identifier de manière unique une fenêtre à l'écran.
Dans Access, chaque formulaire ouvert a une propriété « hWnd » qui renseigne ce descripteur attribué par Windows.

Faites l'expérience
Ouvrez ***fOuSuisJe***.
Enfoncez <Ctrl-key-G> pour atteindre la fenêtre d'exécution (<Ctrl-Key-G>).
Saisissez : ***? Forms!fOuSuisJe.hwnd***
(en français : afficher la propriété hWnd du formulaire ***fOuSuisJe***)
Notez ce numéro.
Fermez la fenêtre pour revenir au formulaire.
Fermez-le et ouvrez-le à nouveau.
Retournez à la fenêtre d'exécution et répétez la commande.
Le numéro a changé !

La fonction va nous donner accès à quatre nombres : *Left*, *Top*, *Right* et *Bottom* qui détermine le rectangle à l'écran.



Précisons : « Top » renseigne la position du premier pixel à l'intérieur de la fenêtre. Par contre, « Bottom » est en dehors, c'est le pixel qui suit le bord inférieur de la fenêtre.
Si l'un vaut ***t*** et l'autre vaut ***b***, alors la hauteur en pixels vaut ***b – t***.
Pareil, pour « Left » (dedans) et « Right »(dehors).

Le coin supérieur gauche de notre formulaire a donc comme coordonnées par rapport au coin supérieur gauche de l'écran :

droite (Right) : Left ;

bas (down) : Top.

Rien n'est simple !

Non seulement ce vocabulaire est déroutant : quand Access dit « droite (*Right*) » Windows dit « Left » ; quand Access dit « Bas (*down*) » Windows dit « Top ».

Mais en plus Windows s'exprime en pixels tandis qu'Access mesure en twips !

Et pour encore corser les choses : le nombre de pixels dépend de la résolution de votre écran !

Il nous faut donc un convertisseur pixels => twips.

En réalité il en faut deux :

- nombre de pixels d'une distance horizontale => twips ;

- nombre de pixels d'une distance verticale => twips.

Mais ne brûlons pas les étapes.

Essayons d'abord de récupérer les données fournies par GetWindowRect.

4.2. Une fonction pour récupérer les données de la position

Voici une fonction qui permet de récupérer, pour un formulaire ouvert dont le nom est donné en paramètre, sa position et ses dimensions exprimées en pixels :

Fonction
```
Option Compare Database
Option Explicit


Type Position
 Droite As Long
 Bas As Long
 Largeur As Long
 Hauteur As Long
End Type



Public Function PositionFormPx(NomDuFormulaire As
String) As Position
Dim rectForm As RECT
'Récupérer le rectangle de la fenêtre du
formulaire
GetWindowRect Forms(NomDuFormulaire).hwnd,
rectForm
'Copier les coordonnées du coin sup gauche
PositionFormPx.Droite = rectForm.Left
PositionFormPx.Bas = rectForm.Top
'Calculer la largeur
PositionFormPx.Largeur = rectForm.Right -
rectForm.Left
'Calculer la hauteur
PositionFormPx.Hauteur = rectForm.Bottom -
rectForm.Top
End Function
```

On commence par définir un ***type de données spécifique***, qui nous permettra de stocker :

- ***Droite*** la distance entre le bord gauche de l'écran et le coin supérieur du formulaire ;

- ***Bas*** la distance entre le bord supérieur de l'écran et le coin supérieur du formulaire ;

- ***Largeur*** de la fenêtre du formulaire ;

- ***Hauteur*** de la fenêtre du formulaire.

Toutes ces données seront exprimées en pixels

Dans la fonction proprement dite, on invoque **GetWindowRect**pour récupérer les coordonnées des quatre coins que l'on exploite pour construire les résultats de la fonction.

Testons tout cela concrètement.

Dans la fenêtre des objets, cliquez sur « Modules » et ensuite sur « Nouveau ».

Dans la fenêtre qui s'ouvre, collez ce code qui correspond à la déclaration de **GetWindowRect :**

```
Declare Function GetWindowRect Lib "user32" ( _
                        ByVal hwnd As Long, _
                        lpRect As RECT) As Long
Type RECT
    Left As Long
    Top As Long
    Right As Long
    Bottom As Long
End Type
```

Sauvegardez en donnant à ce module le nom mAPI (par exemple).

Fermez la fenêtre de l'éditeur VBA.

Dans la fenêtre des objets, cliquez à nouveau sur « Modules » et ensuite sur « Nouveau ».

Dans la fenêtre qui s'ouvre, collez le code de notre fonction **PositionFormPx** :

```
Type Position
 Droite As Long
 Bas As Long
 Largeur As Long
 Hauteur As Long
End Type



Public Function PositionFormPx(NomDuFormulaire As
String) As Position
Dim rectForm As RECT
'Récupérer le rectangle de la fenêtre du
formulaire
GetWindowRect Forms(NomDuFormulaire).hwnd,
rectForm
'Copier les coordonnées du coin sup gauche
PositionFormPx.Droite = rectForm.Left
PositionFormPx.Bas = rectForm.Top
'Calculer la largeur
PositionFormPx.Largeur = rectForm.Right -
rectForm.Left
'Calculer la hauteur
PositionFormPx.Hauteur = rectForm.Bottom -
rectForm.Top
End Function
```

Dans Access, pour en savoir plus sur un mot-clé, placez le curseur de la souris à l'intérieur du mot et enfoncez la touche <F1> : l'aide s'affiche à la bonne page.
Exemple, placez le curseur à l'intérieur de « Type » et enfoncez <F1> : la documentation relative à l'instruction s'affiche.
Si vous utilisez systématiquement cette technique durant votre apprentissage, vos progrès seront spectaculaires.
Ce conseil vaut aussi pour les propriétés des objets (états, formulaires...) : cliquez sur la propriété, elle s'affiche en surbrillance, et enfoncez <F1>.

Dans la barre des menus, cliquez sur « Débogage » et ensuite sur « Compiler... ».

Entre autres, ceci nous permet de vérifier l'absence d'erreur de code.

Sauvegardez en donnant à ce module le nom mFonctions (par exemple).

Fermez la fenêtre de l'éditeur VBA.

Nous allons maintenant constater que la fonction nous donne le résultat attendu.

Modifiez le formulaire ***fOuSuisje*** comme suit :

- propriété « Fen indépendante » => OUI ;
- code de l'événement « Sur ouverture » => ***DoCmd.MoveSize 2835, 3405, 5670, 5100***

Puisque la propriété « Fen indépendante est OUI, le formulaire va se positionner par rapport à l'écran :
à 2835 twips du bord gauche ;
à 3405 twips du bord supérieur ;
avec une largeur de 5670 twips ;
avec une hauteur de 5100 twips.

Affichez le formulaire.

Accédez à la fenêtre d'exécution (<Ctrl-Key-G>) et saisissez :

? PositionFormPx("fOuSuisJe").Droite <Enter> => un nombre s'affiche en dessous.

Répétez avec

? PositionFormPx("fOuSuisJe").Bas <Enter>

? PositionFormPx("fOuSuisJe").Largeur <Enter>

? PositionFormPx("fOuSuisJe").Hauteur <Enter>

Les nombres qui vont s'afficher dépendent de la résolution de votre écran.

Mon écran actuel a une résolution de 1280 x 1024.

J'obtiens respectivement : 189 227 378 340.

Cela veut donc dire que, dans mon environnement :

une distance horizontale de 1 pixel vaut 15 twips (2835 / 189 = 15 ; 5670 / 378 = 15) ;

une distance verticale de 1 pixel vaut 15 twips (3405 / 227 = 15 ; 5100 /340 = 15).

De par sa construction, le rapport twips/pixels est un nombre entier.
Si la résolution de votre écran est différente de celle que j'utilise (1280 x 1024), vous aurez peut-être des rapports différents, voire même avec des décimales !
C'est dû aux arrondis. Vous en comprendrez l'origine plus loin dans ce texte, lorsqu'on affectera des nombres premiers aux paramètres de DoCmd.MoveSize pour voir ce que cela donne.

4.3. Comment déterminer la conversion Pixels => Twips

Avec quelques API, on pourrait calculer les deux coefficients (horizontal et vertical) de conversion Pixels vers Twips.

C'est la solution que choisiront les virtuoses, entre autres les deux auteurs cités plus haut.

Nous allons utiliser une méthode plus artisanale, cela nous permettra de mettre à profit ce que nous savons déjà faire :

- nous pouvons localiser un formulaire ouvert à l'écran et trouver ses dimensions en nous exprimant en pixels, c'est notre fonction *PositionFormPx* ;
- d'autre part, depuis la version Access2000 un formulaire dispose de deux propriétés ***WindowHeight*** (HauteurFenêtre) et ***WindowWidth*** (LargeurFenêtre) toutes deux exprimées en twips.

Dès lors, si nous affichons notre formulaire et que dans la foulée, nous récupérons :
- sa **largeur** avec *PositionFormPx(NomDuFormulaire).Largeur ;*
- ainsi que sa **hauteur** avec *PositionFormPx(NomDuFormulaire).Hauteur,*
toutes deux exprimées en pixels, nous disposerons ainsi de deux longueurs (une horizontale et une verticale) exprimées dans les deux unités de mesure... et nous pourrons calculer leur rapport de conversion.

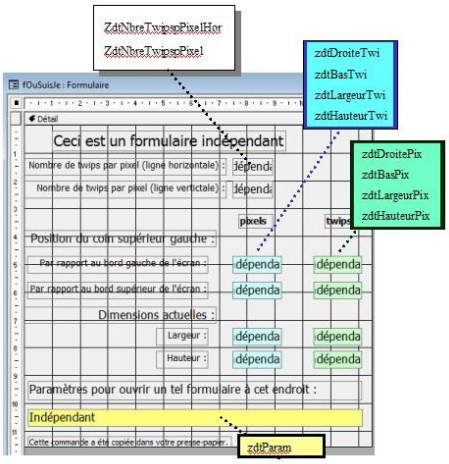
Nous traduirons ceci en code dans le paragraphe suivant.

5. Un formulaire étalon pour trouver les paramètres de doCmd.MoveSize

Nous allons utiliser tout ceci dans notre formulaire ***fOuSuisJe***.

5.1. Les contrôles du formulaire

Garnissons-le de quelques étiquettes et zones de texte. Pour rester cohérent avec la suite des explications, veuillez baptiser les zones de texte comme indiqué ci-après :



Nos zones de texte n'ont pas de source. Nous allons les garnir par programmation.

Sauvegardons notre travail : <Ctrl-Key-S>.

5.2. Le code pour calculer le nombre de twips par pixel

Comme dit plus haut, lors de l'ouverture du formulaire, nous allons calculer le nombre de twips par pixel horizontal en divisant sa largeur exprimée en twips par sa largeur en pixels.

Nous procéderons de même avec sa hauteur pour déterminer le nombre de twips par pixel vertical.

Voici le code pour effectuer ces actions :

```
Me.ZdtNbreTwipspPixelHor = Me.WindowWidth /
PositionFormPx(Me.Name).Largeur
```

En français : on divise la largeur en twips par la largeur en pixels et on loge le résultat dans le contrôle ZdtNbreTwipspPixelHor.

De même :

```
Me.ZdtNbreTwipspPixelVer = Me.WindowHeight /
PositionFormPx(Me.Name).Hauteur
```

On divise la hauteur en twips par la hauteur en pixels et on loge le résultat dans le contrôle ZdtNbreTwipspPixelVer.

Nous allons déclencher l'exécution de ce code dans l'événement « Sur ouverture » de notre formulaire.

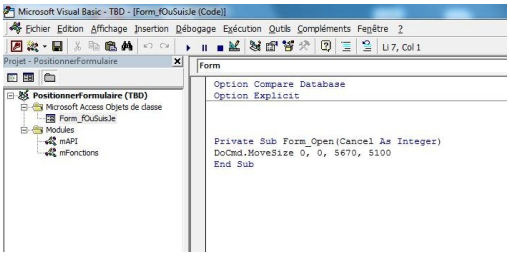
Accédez au code actuel :

- <Alt-key-F11> pour ouvrir l'éditeur de code VBA ;

- <Ctrl-key-R> pour afficher l'explorateur de projets ;

- double-cliquez sur Form_fOuSuisJe ;

Vous devriez avoir quelque chose comme ceci à l'écran :Dès lors, si nous affichons notre formulaire et que dans la foulée, nous récupérons :



Remplacez le code actuel par celui-ci :

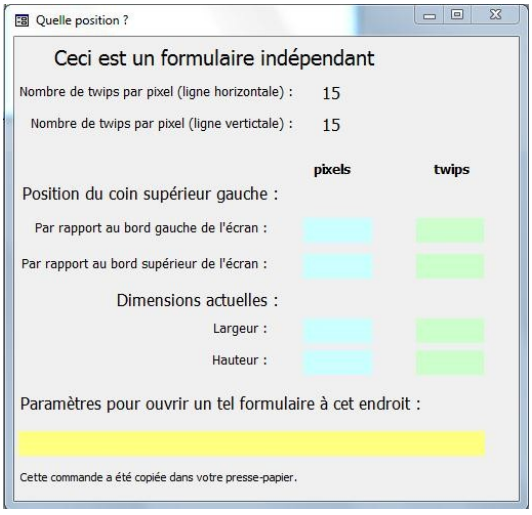
```
Option Compare Database
Option Explicit



Private Sub Form_Open(Cancel As Integer)
'Calculer le Nbre de twips par Pixel
    'Diviser la largeur en Twips (Me.WindowWidth) par la largeur en Pixels (PositionFormPx)
    Me.ZdtNbreTwipspPixelHor = Me.WindowWidth / PositionFormPx(Me.Name).Largeur
    'Diviser la Hauteur en Twips (Me.WindowHeight) par la hauteur en Pixels (PositionFormPx)
    Me.ZdtNbreTwipspPixelVer = Me.WindowHeight / PositionFormPx(Me.Name).Hauteur
End Sub
```

Pour ***WindowWidth*** et ***WindowHeight***, rappelez-vous

Refermez la fenêtre de l'éditeur de code.

Fermez le formulaire en le sauvegardant.

Ouvrez-le à nouveau et vous obtenez quelque chose comme ceci :



5.3. Le code pour attribuer une valeur à chaque zone de texte

Complétons l'habillage des autres contrôles.

Mémorisons dans *zdtDroitePix* et *zdtBasPix* les coordonnées en pixels du coin supérieur gauche à l'aide des valeurs de notre fonction PositionFormPx :

```
Me.zdtDroitePix = PositionFormPx(Me.Name).Droite
Me.zdtBasPix = PositionFormPx(Me.Name).Bas
```

Et dans *zdtDroiteTwi* et *zdtBasTwi* la conversion en twips en multipliant les précédents par leur coefficient multiplicateur :

```
Me.zdtDroiteTwi = Me.zdtDroitePix * Me.ZdtNbreTwipspPixelHor
Me.zdtBasTwi = Me.zdtBasPix * Me.ZdtNbreTwipspPixelVer
```

Procédons par analogie, pour les largeur et hauteur :

```
Me.zdtLargeurTwi = Me.WindowWidth
Me.zdtHauteurTwi = Me.WindowHeight
Me.zdtLargeurPix = Me.zdtLargeurTwi / Me.ZdtNbreTwipspPixelHor
Me.zdtHauteurPix = Me.zdtHauteurTwi / Me.ZdtNbreTwipspPixelVer
```

Pour la zone de texte zdtParam, il suffit de coller les morceaux :

La commande

```
Me.zdtParam = "DoCmd.MoveSize " & Me.zdtDroiteTwi & " , " & Me.zdtBasTwi & " , " _
                                & Me.zdtLargeurTwi & " , " & Me.zdtHauteurTwi
```

Et pour que l'utilisateur puisse récupérer cette commande dans le presse-papier, on programme les opérations qu'il ferait manuellement : sélectionner le texte et copier

Commande dans le presse-papier

```
Me.zdtParam.SetFocus
'on sélectionne tout le texte
Me.zdtParam.SelStart = 0
Me.zdtParam.SelLength = Len(Me.zdtParam.Text)
'on copie dans le presse-papier
DoCmd.RunCommand acCmdCopy 'équivalent du raccourci Ctrl-Key-C
```

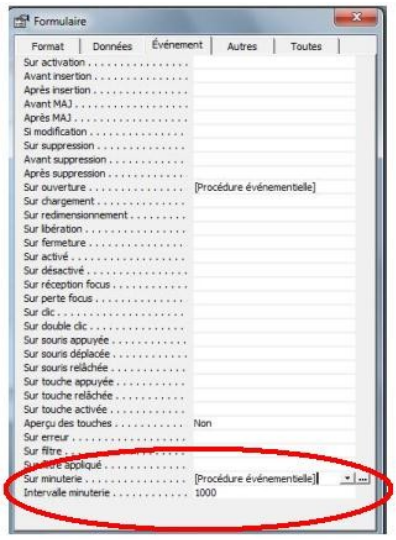
5.4. Quand déclencher le processus ?

L'idée est que le formulaire affiche ses données instantanément quelles que soient la position et les dimensions choisies par l'utilisateur qui le déplace et le redimensionne à sa guise.

Nous allons faire en sorte que le code se déclenche par exemple toutes les secondes.

Affichons les propriétés du formulaire , onglet « Événement »,

fixons à 1000 (millisecondes) l'intervalle de minuterie et double-cliquons sur « Sur minuterie » :



Un clic sur les « ... » pour ouvrir l'éditeur de VBA, où nous insérons notre code :

Événement sur minuterie

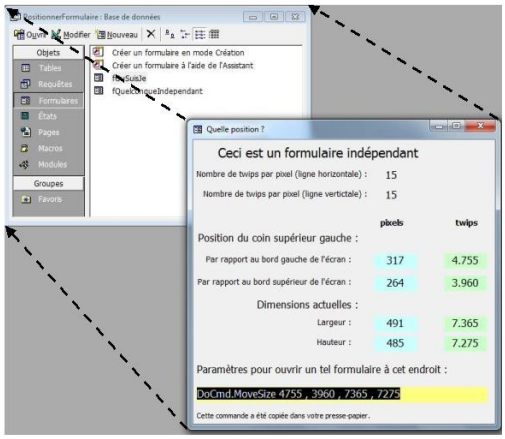
```
Private Sub Form_Timer()
'La position du coin sup gauche
Me.zdtDroitePix = PositionFormPx(Me.Name).Droite
Me.zdtBasPix = PositionFormPx(Me.Name).Bas
Me.zdtDroiteTwi = Me.zdtDroitePix *
Me.ZdtNbreTwipspPixelHor
Me.zdtBasTwi = Me.zdtBasPix *
Me.ZdtNbreTwipspPixelVer
'La Largeur et la hauteur
Me.zdtLargeurTwi = Me.WindowWidth
Me.zdtHauteurTwi = Me.WindowHeight
Me.zdtLargeurPix = Me.zdtLargeurTwi /
Me.ZdtNbreTwipspPixelHor
Me.zdtHauteurPix = Me.zdtHauteurTwi /
Me.ZdtNbreTwipspPixelVer
Me.zdtParam = "DoCmd.MoveSize " & Me.zdtDroiteTwi
& " , " & Me.zdtBasTwi & " , " _
                    & Me.zdtLargeurTwi & " ,
" & Me.zdtHauteurTwi
'Commande dans le presse-papier
Me.zdtParam.SetFocus
'on sélectionne tout le texte
Me.zdtParam.SelStart = 0
Me.zdtParam.SelLength = Len(Me.zdtParam.Text)
'on copie dans le presse-papier
DoCmd.RunCommand acCmdCopy 'équivalent du
raccourci Ctrl-Key-C
End Sub
```

Sauvegardez le formulaire par précaution. Ouvrez-le, déplacez-le, redimensionnez-le : les données s'actualisent.

5.5. La preuve que ça marche

Faites l'expérience :

Positionnez ***fOuSuisJe*** de manière telle qu'il recouvre exactement la surface de la fenêtre des objets.



Votre presse-papier contient donc la syntaxe de la commande.

Fermez ***fOuSuisJe***.

Dans un formulaire **indépendant** quelconque, dans l'événement sur ouverture, collez le code qui se trouve dans votre presse-papier (<Ctrl-Key-V>).

Ouvrez ce formulaire : il recouvre exactement la fenêtre des objets.

Cqfd.

5.6. Si ce gadget vous intéresse

Vous pouvez l'importer dans votre base de données, par exemple comme ceci :

vous affichez côte à côte à l'écran la base de données jointe et la vôtre ;

dans *PositionnerFormulaire.mdb* vous mettez en surbrillance le module mAPI et avec le bouton gauche de la souris enfoncé, vous faites glisser dans la fenêtre de votre application ;

vous procédez de même pour mFonctions et fOuSuisJe... et c'est à vous !

6. Une dernière pour la route...

Quand nous avons élaboré la fonction pour déterminer la conversion Pixels => Twips, j'avais attiré votre attention

sur le fait qu'Access ne positionne pas nécessairement exactement, à droite et en bas, au nombre de twips mentionnés dans la commande ***DoCmd.MoveSize***. Chaque paramètre est arrondi au plus proche multiple de twips par pixel.

Notre ***fOUSuisJe*** nous permet de constater le phénomène.

Dans l'événement « Sur ouverture », placez cette instruction en tête :

DoCmd.MoveSize 2887, 5167, 10267, 8269

Quelle que soit la résolution de votre écran, donc votre nombre de twips par pixel, vous constaterez que le ***fOuSuisJe*** affiché n'a aucun paramètre qui coïncide avec ceux de la commande. En effet, 2887, 5167, 10267 et 8269 sont des nombres premiers, ils ne sont donc pas des multiples de vos rapports twips/pixels.

À titre d'exemple, avec ma résolution 1280 x 1024, j'ai en réalité : 2880, 5160, 10260 et 8265 qui sont les multiples de 15 les plus proches des paramètres originaux.

7. Conclusion

Ceci est une première étape.

Un tutoriel suivra dans lequel on mettra en pratique la technique qui vient d'être exposée :

- positionner un formulaire par rapport à un contrôle d'un autre formulaire ;
- rendre deux formulaires solidaires : le déplacement de l'un, entraînant le déplacement de l'autre.

Cela nous permettra d'évoquer les propriétés ***WindowLeft*** et ***WindowTop*** disponibles seulement à partir de Access2002 mais nous décrirons un moyen de pallier leur absence en Access2000.

Vous pouvez télécharger la base de données qui a servi d'exemple : Lien 70.

Retrouvez l'article de Claude Leloup en ligne : Lien 71.

Vidéo Excel 2010 : comparer des listes grâce à la mise en forme conditionnelle

Cette vidéo illustre la manière de signaler les éléments d'une liste non présents dans une autre. Au départ d'une liste existante, je vous détaille les étapes à réaliser pour créer un outil générique de comparaison basé sur une Mise en Forme Conditionnelle (MFC) personnalisée.

1. Introduction

1.1. Objectifs

Cette vidéo formative poursuit les objectifs suivants :

- illustrer l'utilisation des tableaux 2010 ;
- maîtriser la création d'une mise en forme conditionnelle basée sur le résultat d'une formule ;
- mettre en lumière l'utilité d'une conception professionnelle d'un classeur Excel.

Ces objectifs seront approchés par la création de deux tableaux de données, l'un permettant la saisie de données dont on veut vérifier la présence ou l'absence dans un deuxième tableau.

L'exemple utilisé dans la vidéo montrera comment mettre en évidence des articles qui n'ont pas été vendus durant une certaine période. L'on disposera donc d'une liste de ventes réalisées et d'un tableau comparatif dans lequel seront reprises les références d'articles. Les articles de cette liste qui n'ont pas été vendus seront affichés automatiquement dans un format spécifique grâce à une mise en forme conditionnelle personnalisée.

Dans la vidéo, j'ai choisi de mettre en évidence les articles non vendus, mais il suffira de modifier l'opérateur de comparaison pour afficher les articles vendus, voire uniquement ceux qui ont été vendus un certain nombre de fois, etc.

C'est votre imagination et vos "besoins métier" qui vous feront adapter l'exemple illustré aux cas que vous devrez résoudre.

On pourrait donc, grâce à cet exercice, vérifier :

- la présence ou l'absence de membres du personnel dans une liste de pointage ;
- la présence ou l'absence de certains types d'incidents dans une liste de non-conformités rencontrées ;
- ...

La "complexification" de la formule conditionnelle vous permettra de résoudre divers problèmes liés à la comparaison de données au sein de listes de données.

Comme toujours, la bonne conception d'un classeur et l'utilisation d'outils adéquats tels qu'illustrés dans la vidéo, tant pour la forme (affichage, multifenêtre) que le fond (tableaux, formules, fonctions, mise en forme conditionnelle) garantiront une pérennité et une évolutivité des classeurs.

1.2. Prérequis

Maîtriser les **références absolues et relatives** (utilisation du signe $ dans l'adressage d'une référence) est le minimum requis. L'utilisateur mal à l'aise avec ces notions gagnera à lire mon tutoriel sur le sujet : Lien 72.

Dans ce tutoriel, nous utiliserons une formule requérant la manipulation de fonctions Excel (ET() et NB.SI()). Comprendre la création d'une formule et l'utilisation de fonctions basiques sera évidemment utile à l'exploitation de cette vidéo.

1.3. Niveau

A priori, cette vidéo s'adresse à des personnes ayant des bases en Excel et voulant progresser.

Les utilisateurs d'anciennes versions d'Excel (2003 et antérieures) trouveront ici des idées d'utilisation des nouveaux outils disponibles depuis la version 2007.

1.3.1. Durée

Cette vidéo d'une durée de 14 minutes est découpée en huit chapitres techniques, vous donnant ainsi la possibilité de reprendre la vidéo sur une thématique que vous souhaitez revoir.

2. La vidéo

Lien 73.

3. Conclusions

La vidéo illustre qu'il n'est guère compliqué de manipuler différents outils d'Excel (tableaux, formules, mise en forme conditionnelle) pour mettre en place rapidement des classeurs pérennes et professionnels.

Une maîtrise globale de ce que peut faire Excel alliée à une conception rigoureuse de vos classeurs vous permettra de réaliser des outils performants aptes à vous aider dans la gestion, et surtout l'analyse, des données avec Excel.

N'hésitez pas à me faire part de vos remarques et suggestions, par rapport à la problématique posée en début de tutoriel. Si vous souhaitez un tutoriel vidéo sur un sujet précis de l'utilisation d'Excel, vous pouvez bien entendu m'en faire part.

Bon travail avec Excel !

Retrouvez l'article de Pierre Fauconnier en ligne : Lien 74.

Import et export des fichiers Microsoft dans l'environnement SAS 9.2 sur Windows 64-bits

Identification des problèmes et solutions envisageables pour l'import et l'export des fichiers Microsoft Excel ou Access dans l'environnement SAS 9.2 sur Windows 64-bits.

1. Introduction

Avec l'arrivée des OS Windows 64-bits, de nombreux problèmes d'importation et d'exportation de fichiers Office sont apparus sur SAS. En effet, lorsque l'on souhaite importer des fichiers Excel ou Access dans les environnements Windows 64-bits (Windows 2008 64-bits, Windows 2008 R2, Windows Vista 64 et Windows 2003 64-bit) ce message est susceptible d'apparaître lorsque l'on utilise un code développé initialement sur les plates-formes 32-bits :

ERROR: DBMS type EXCEL (ACCESS) not valid for import.

Le problème est récurrent, car la stratégie de la plupart des sociétés étant d'installer la version 64-bits de SAS sur des serveurs ou des PC, l'application n'est alors plus compatible avec le moteur de connexion DBMS=EXCEL qui est un moteur 32-bits.

Si cette stratégie est conservée, il est nécessaire d'utiliser les moteurs de connexion 64-bits tels qu'ils sont décrits dans les exemples ci-dessous et de connaître les incompatibilités entre les commandes SAS, les moteurs et les types de fichiers Office. Autrement, il est possible d'installer la version 32-bits de SAS sur la plate-forme 64-bits et d'utiliser les connecteurs habituels.

Ce dossier technique présente les différents changements qui doivent être pris en compte pour importer et exporter les fichiers Office. Il a été préparé à partir du retour d'expérience des missions de conseil réalisé par l'auteur chez ses clients et des discussions avec certains experts chez l'éditeur et le support. Bien entendu, l'auteur présente par avance ses excuses si d'autres cas réels n'ont pas été appréhendés dans ce dossier alors qu'ils auraient pu aider le lecteur.

La première partie de cet article présente les possibilités d'accès aux fichiers en utilisant le client traditionnel de SAS sur la plate-forme SAS 9.2 64-bits.

La seconde partie détaille les erreurs courantes rencontrées lors de l'utilisation des nouvelles syntaxes fournies détaillées auparavant.

La dernière partie donne des solutions pour stabiliser le système d'échange entre SAS et Office et présente la façon dont SAS Enterprise Guide gère l'import et l'export de données.

2. Accès aux fichiers sur la plate-forme SAS 9.2 64-bits

Dans cette première partie, nous allons nous intéresser aux deux modules d'accès aux données Office (Excel et Access) présents dans SAS : le module SAS ACCESS TO PC FILES et le module SAS PC FILE SERVER.

Tout d'abord, une présentation du nouveau module SAS PC FILE SERVER sera donnée. Les moteurs inclus dans ce module permettent de gérer les accès aux données sur les plates-formes UNIX et Windows 64-bits.

Ensuite, nous étudierons grâce à des exemples concrets le moteur de connexion à des données Excel 2003 et 2007, puis sur le même mode travail, nous présenterons le moteur de connexion à des données Access 2003 et 2007.

Les exemples détaillés de cette partie concerneront tout aussi bien l'importation de données que l'exportation de celles-ci, au travers des procédures et du LIBNAME.

Les données utilisées n'ont rien de spécifique à ce dossier. L'ensemble des tests a été réalisé grâce aux tables de la bibliothèque SASHELP. Ainsi, les données de la table SASHELP.CLASS ont été copiées dans un fichier Excel 2003 (nommé D_Excel_2003.xls) puis 2007 (nommé D_Excel_2007.xlsx) et ensuite dans un fichier Access 2003 (nommé D_Access_2003.mdb) et 2007 (nommé D_Access_2007.accdb).

2.1. La nouvelle solution SAS PC FILE SERVER

PC FILE SERVER est une application qui gère les requêtes de lecture et d'écriture de fichiers Office. Elle est fournie en standard pour des plates-formes UNIX et Windows 64-bits dans les dépôts SAS.

Elle propose deux moteurs de connexion, EXCELCS et ACCESSCS, permettant de répondre à des besoins précis que nous verrons plus loin. Installée en tant que service à partir des plans dans le dépôt SAS, elle utilise le Microsoft Access Connectivity Engine et les drivers ODBC version 12 pour Microsoft Excel et Microsoft Access.

Par extension, cette application permet d'accéder également à des SGBD ayant des connecteurs ODBC, mais ce point ne sera pas abordé dans cet article.

2.2. Les moteurs de connexion envisageables

Avec l'ajout de cette solution, il est possible de gérer les différentes versions des fichiers Office sur la plate-forme 64-bits de Windows avec différents moteurs :

- la source de données MS-Excel amène

l'utilisation des moteurs SAS suivants : « XLS », « EXCELCS », « PCFILES » ;
- la source de données MS-Access amène l'utilisation des moteurs SAS suivants : « ACCESSCS », « PCFILES ».

Mais pourquoi avoir autant de moteurs ? Se valent-ils en performance ? Ce dossier amènera les réponses à ces questions.

2.3. Connexion avec le logiciel Excel

Sur une plate-forme 64-bits, il existe trois moteurs de connexion qui possèdent leurs propres contraintes.

Le moteur « XLS »
- Ne peut pas communiquer avec les fichiers Microsoft Office 2007 files (XLSX, XLSM, et XLSB).
- Ne peut pas être utilisé avec un LIBNAME. Il ne peut être utilisé qu'au sein des PROC IMPORT et EXPORT.
- Reste le moteur le plus stable pour tout autre type de fichiers (i.e. hormis Excel 2007).

Le moteur « EXCELCS »
- Peut communiquer avec les fichiers Microsoft Office 2007 files (XLSX, XLSM, et XLSB).
- Ne peut pas être utilisé avec un LIBNAME. Il ne peut être utilisé qu'au sein des PROC IMPORT et EXPORT.
- Ne renomme pas les colonnes ayant des caractères spéciaux (il remplace ces caractères par des soulignements alors que le moteur XLS renomme complètement la variable en VAR*n*).
- Peut créer des fichiers Excel 2007. Toutefois, s'il s'agit de créer un fichier au format Excel 2007 plutôt que de mettre à jour un fichier existant, il faut lui donner une extension XLSB (B pour Binaire).

Le moteur « PCFILES »
- Peut communiquer avec les fichiers Microsoft Office 2007 (XLSX, XLSM, et XLSB).
- Doit être utilisé avec un LIBNAME. Il ne peut pas être utilisé au sein des PROC IMPORT et EXPORT.
- L'utilisation de PCFILES est basée sur les moteurs (engine) Microsoft : Microsoft Jet Engine et Microsoft ACE engine. Par défaut le LIBNAME utilise MSENGINE= ACE.

Nous allons détailler ci-dessous quelques exemples d'importation et d'exportation à des données Excel 2003 (*.xls) et 2007 (*.xlsx) en utilisant SAS 9.2 64-bits. Nous allons comparer les résultats des moteurs du module SAS PC FILE SERVER avec les moteurs du module ACCESS to PC FILES (classiquement utilisé dans les versions 32-bits de SAS).

Pour cela, les procédures IMPORT et EXPORT seront présentées avec quelques options spécifiques, à savoir :
- GETNAMES : option permettant de récupérer les noms de colonnes dans la première ligne du fichier Excel ;
- SHEET : option permettant de spécifier une feuille particulière du classeur Excel, tant pour l'import que pour l'export ;
- MSENGINE : option permettant de changer le moteur de connexion fourni par Windows ;
- REPLACE : option nécessaire pour remplacer le fichier en sortie d'une procédure.

Ensuite, l'étape LIBNAME sera utilisée pour illustrer ce type de connexion.

2.3.1. Utilisation de la PROC IMPORT

Voir tableau en ligne : Lien 75

2.3.2. Utilisation du LIBNAME

Voir tableau en ligne : Lien 76

2.3.3. Utilisation de la PROC EXPORT

Les résultats de ces exécutions valent pour la création d'un fichier ou pour l'ajout d'un onglet dans un classeur existant.

À noter également, lorsque l'export doit écraser un fichier existant, un backup est créé sous la forme d'un fichier ayant une extension .bak après l'extension .xls (par exemple : monfichier.xls.bak)

Voir tableau en ligne : Lien 77

2.4. Connexion avec le logiciel Access

Sur une plate-forme 64-bits, il est obligatoire d'utiliser le moteur ACCESSCS pour gérer les accès dans les procédures (PROC). Les affectations de bibliothèques (LIBNAME) utiliseront le moteur PCFILES.

Le moteur ACCESSCS :
- peut lire les fichiers Microsoft Office 2007 (ACCDB) ;
- ne peut pas être utilisé avec un LIBNAME. Il ne peut être utilisé qu'au sein des PROC IMPORT et EXPORT ;
- gère correctement les noms de colonnes ayant plusieurs caractères spéciaux.

Le moteur PCFILES :
- peut lire les fichiers Microsoft Office 2007 (ACCDB) ;
- doit être utilisé avec un LIBNAME. Il ne peut pas être utilisé au sein des PROC IMPORT et EXPORT ;
- l'utilisation de PCFILES est basée sur les moteurs (engine) Microsoft : Microsoft Jet Engine et Microsoft ACE engine. Par défaut le LIBNAME utilise MSENGINE= ACE.

Nous allons détailler ci-dessous quelques exemples d'importation et d'exportation à des données Access 2003 (*.mdb) et 2007 (*.accdb) en utilisant SAS 9.2 64-bits. Nous allons comparer les résultats des moteurs du module SAS PC FILE SERVER avec le moteur du module ACCESS to PC FILES (classiquement utilisé dans les

versions 32-bits de SAS).

Pour cela, les procédures IMPORT et EXPORT seront présentées avec quelques options spécifiques, à savoir :

- SHEET : option permettant de spécifier une feuille particulière du classeur Excel, tant pour l'import que pour l'export ;
- REPLACE : option nécessaire pour remplacer le fichier en sortie d'une procédure.

Ensuite, l'étape LIBNAME sera utilisée pour illustrer ce type de connexion.

2.4.1. Utilisation de la PROC IMPORT

Pour importer une table de la base Access, 2003 ou 2007, il convient d'utiliser uniquement le moteur ACCESSCS.

Voir tableau en ligne : Lien 78.

2.4.2. Utilisation du LIBNAME

La seule façon de créer un LIBNAME sur une base Access reste la solution employant le moteur PCFILES.

Voir tableau en ligne : Lien 79.

2.4.3. Utilisation de la PROC EXPORT

Les tests confirment qu'un seul moteur est exploitable sur cette plate-forme. Le moteur habituel "Access" ne fonctionne plus.

Voir tableau en ligne : Lien 80.

3. Erreurs couramment rencontrées avec le module PCFILE SERVER

Ce chapitre présente les erreurs rencontrées au fil du temps dans l'usage de ces connecteurs sur la plate-forme 64-bits. Ces erreurs sont non imputables au code SAS mais plutôt à la stabilité du système et des composants SAS.

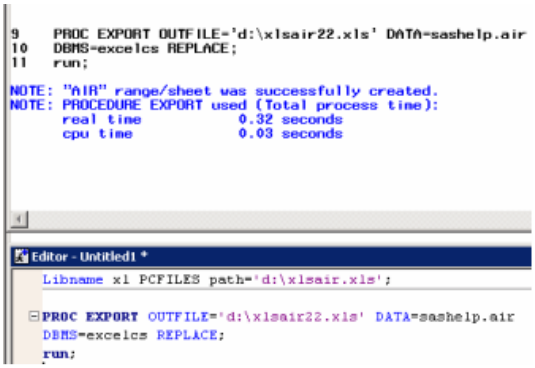
3.1. Mauvaise initialisation du service PCFILE SERVER

3.1.1. Erreur sur les procédures

Problème : Il arrive que la connexion n'ait pas le temps de s'initialiser correctement, notamment lors des PROC EXPORT.

```
1     PROC EXPORT OUTFILE='d:\xlsair.xls'
DATA=sashelp.air
2     DBMS=Excelcs REPLACE;
3     run;

ERROR: Error attempting to CREATE a DBMS table.
ERROR: CLI execute error: Server communication
failure.  The client log file (PCFILES.log) may
contain additional  information..
ERROR: CLI disconnect failed: Server
communication failure.  The client log file
(PCFILES.log) may contain additional information.
ERROR: CLI disconnect failed: Server
communication failure.  The client log file
(PCsFILES.log) may contain additional
information.
ERROR: Error in the LIBNAME statement.
ERROR: Export unsuccessful.  See SAS Log for
details
```

Solution : comme le montre l'exemple de code suivant, il faut créer avant la PROC EXPORT un LIBNAME sur le fichier de destination qui pourtant n'existe pas encore.



3.1.2. Erreur sur les LIBNAME

Problème : lors de l'utilisation d'un LIBNAME, ici xls_obj, il arrive que la connexion n'ait pas le temps de s'initialiser correctement.

sas

```
data EXTRACTION (drop=CENTRE CODE_CENTREPRODUIT
2011-04-01T07:45:23,871 INFO  [00000015] - 612!
rename=(centre2=centre id_centre=CODE_CENTRE
PRODUIT2=PRODUIT ));
2011-04-01T07:45:23,871 INFO  [00000015] - 613
set xls_obj.OBJECTIFS_RM;


ERROR: CLI prepare error: Server communication
failure.  The client log file (PCFILES.log) may
contain additional information.
```

Solution : lorsqu'il s'agit d'accéder à des fichiers Excel 2000-2003, il est préférable d'utiliser l'option MSENGINE= JET afin de minimiser ce genre d'erreur.

```
LIBNAME xls_obj PCFILES path=
"\ref\OBJECTIFS_RM.xls" MSENGINE= JET ;
```

L'utilisation de PCFILES est basée sur les moteurs Microsoft : Microsoft Jet Engine et Microsoft ACE Engine. Par défaut le LIBNAME utilise MSENGINE= ACE.

3.2. Impossibilité d'atteindre un fichier sur le réseau

Problème : l'erreur suivante provient encore de l'interaction de PC FILE SERVER avec le système.

```
Libname xl2007 PCFILES
"SasData\D_Excel_2007.xlsx";
```

```
ERROR: CLI error trying to establish connection:
[Microsoft] [ODBC Excel Driver] '(unknown)' is
not a valid path.  Make sure that the path name
is spelled correctly and that you are connected
to the server on which the file resides.
ERROR: Error in the LIBNAME statement. Connection
Failed.
See log for details.
```

Solution : lorsque le fichier à importer ou à exporter se trouve sur un lecteur réseau (i.e. en dehors des lecteurs physiquement rattachés à la machine, soit dans 95 % des

cas maintenant) il arrive que PC FILE SERVER en tant que service Windows n'ait pas pu reconnaître le mappage réseau (ce dernier étant déclaré après le démarrage des services).

Dans ce cas, il faut spécifier le nom universel du chemin du fichier, soit redémarrer le service après que le mappage réseau a été réalisé.

Le nom universel (UNC, *Universal Naming Convention* en anglais) est le nom du répertoire composé de la façon suivante : *\\nom_du_serveur\Nom_du_repertoire\D_Excel_2007.xlsx*

3.3. Impossibilité de remplacer un fichier existant

Cette erreur est susceptible d'apparaître lorsque de nombreuses PROC EXPORT sont générées par une boucle MACRO (%do %end). Le système ne semble plus suivre et refuse d'accéder aux fichiers.

Dans ce cas, plutôt que de laisser SAS tenter de remplacer le fichier cible, il est préférable de le supprimer soi-même. La méthode la plus simple étant de passer la commande SAS suivante juste avant la PROC EXPORT.

Dans l'exemple, elle permet de supprimer physiquement le fichier nommé mybook.xls :

```
data _null_;
fname="fictmp";
rc=filename(fname,"D:\temp\mybook.xls");
if rc=0 and fexist(fname) then rc=fdelete(fname);
rc=filename(fname);
run;
```

Note : il peut arriver dans certains cas que le fichier ne puisse même plus être supprimé. L'expérience a montré que l'utilisateur SYSTEM se retrouvait propriétaire du fichier et ne pouvait rendre les droits au propriétaire originel : si l'étape DATA présentée ci-avant est exécutée par l'utilisateur, elle ne peut fonctionner car il ne sera plus le propriétaire du fichier.

Il faut alors redémarrer le service PC FILE SERVER et se connecter en tant qu'administrateur pour libérer les fichiers. Ce cas est fort heureusement peu courant.

4. Les solutions à apporter pour stabiliser ou réussir ses échanges entre SAS et les produits Office

4.1. Installer une version récente de SAS 9.2

Installer SAS 9.2 Maintenance 3 permet de profiter des derniers correctifs. Cette version a été largement améliorée au niveau de la stabilité et supporte la plupart des plates-formes 64-bits.

SAS 9.3 M0 n'a pas changé la gestion des drivers Excel, ce ne peut être la raison de migrer malheureusement.

4.2. Installer une version d'Excel compatible

La dernière version mise à jour par MS Office pour Excel 2007 est la version 12.0.6557, le support SAS liste dans cette note : Lien 81, quelques problèmes pouvant encore apparaître lors de l'utilisation de celle-ci.

Pour éviter tout problème de stabilité entre SAS et Excel 2007, il vaut mieux utiliser la version 12.0.4518. Autrement, les utilisateurs qui vont ouvrir des fichiers Excel depuis le serveur (lorsque les utilisateurs travaillent directement sur le serveur depuis MSTSC par exemple), risqueront de visualiser un fichier avec de nombreuses données manquantes alors que ce même fichier peut être ouvert sur leur poste (XP 32-bits) sans problème.

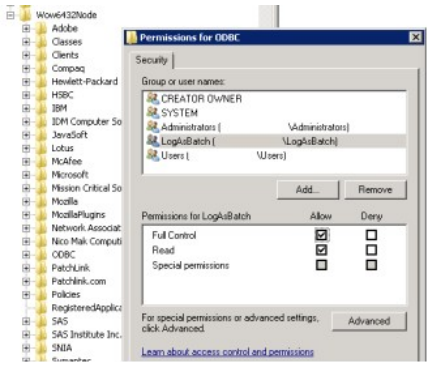
4.3. Installer les drivers spécifiques Microsoft Access Database Engine 2010

Ces drivers semblent plus stables pour les échanges. Indépendamment du niveau de maintenance de SAS 9.2 (M1, M2, M3).

http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=13255

4.4. Donner les pleins droits aux utilisateurs sur une clef de registre

Afin d'éviter certaines erreurs du service PC FILE SERVER, il est parfois nécessaire de modifier les droits sur HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\ODBC en donnant les pleins droits au groupe SAS. Dans l'exemple ci-dessous le groupe SAS se nomme LogAsBatch et hérite des pleins droits sur la clef ODBC.



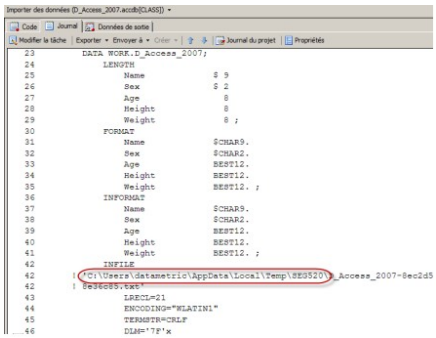
4.5. Utiliser SAS Enterprise Guide

La version SAS Enterprise Guide (EG) 4.1 possède quelques bogues : le plus connu étant le blocage des futurs imports d'un fichier sur le nombre d'observations issu du premier import. Par exemple, si le fichier a été importé à l'instant avec vingt lignes, une seconde importation n'en importera pas plus même si le fichier Excel a été modifié entre-temps (cf. Lien 82).

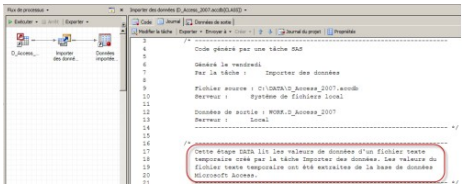
Néanmoins, si l'on se focalise sur la version EG 4.2 et 4.3, il arrive très souvent que l'utilisateur de SAS Enterprise Guide (EG) constate un meilleur fonctionnement dans ces imports/exports qu'avec le client traditionnel.

À la différence du client traditionnel, SAS Enterprise Guide n'utilise pas le module SAS ACCESS TO PC FILES. SAS EG réalise ces importations et exportations en deux temps. Dans un premier temps, il utilise un composant Windows pour transformer les fichiers Excel ou Access en un fichier texte stocké dans un espace temporaire (dans l'image illustrant cette action, le répertoire est C:\Users\datametric\AppData\Local\Temp\

SEG4020).



Dans un second temps, l'assistant d'importation crée une étape DATA pour récupérer le résultat. L'image montre bien que SAS crée une étape DATA basée sur l'INFILE.



Il faut donc conclure que la tâche d'import de SAS visible dans le projet n'est donc que la phase finale du processus d'importation. Il en va de même pour la tâche d'exportation, l'outil va utiliser une API pour créer les fichiers au format XLS et MDB.

En ce qui concerne les exports au format XLSX, EG 4.3 n'a pas la possibilité d'exporter nativement dans ce format. Il faut utiliser cette tâche personnalisée mise à disposition par SAS :

- http://support.sas.com/documentation/onlinedoc/guide/customtasks/samples/SAS.Tasks.ExcelExport.zip

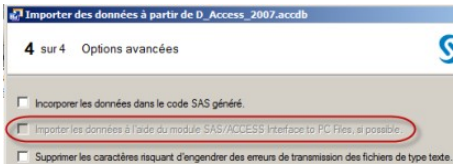
Une fois de plus, cet outil n'utilise pas le module SAS mais les composants Windows.

Sachez que SAS Enterprise Guide pourrait utiliser ce module et donc générer des PROC IMPORT et EXPORT réexploitables. Cependant, les conditions d'utilisation sont strictes et le code généré utilisera le moteur de connexion DBMS=EXCEL ou ACCESS.

Les conditions sont les suivantes :

- le fichier Excel doit être stocké là où se trouve le moteur d'exécution (si SAS est exécuté en local alors le fichier doit être en local ou sur un répertoire visible depuis l'explorateur Windows) ;
- si le moteur DBMS=EXCEL ou ACCESS n'est pas valide (comme sur les plates-formes 64-bits), l'outil ne permettra pas d'activer cette option.

L'image suivante montre que l'assistant désactive l'option sur une plate-forme 64-bits.



5. Conclusion

Afin d'utiliser les procédures d'import et d'export de SAS 9.2 sur une plate-forme 64-bits, il est nécessaire de connaître les différents moteurs de connexion ainsi que leurs impacts sur les fichiers de résultat. Les exemples fournis plus haut permettent de se faire une idée des erreurs qui peuvent être générées et des options utilisables ou non.

Dans le cas d'une migration d'une plate-forme Windows 32-bits à 64-bits, les programmes devront être modifiés sur ces points, même si l'ensemble paraît s'intégrer dans une migration à isopérimètre.

Ce tableau synthétise les possibilités d'utilisation des moteurs de connexion selon la source et l'utilisation d'une Procédure ou d'un LIBNAME.

Pour les fichiers Excel 2003

Nom du moteur	Excel	xls	ExcelCS	PCFILES
LIBNAME	Non	Non	Non	Oui
Proc IMPORT	Non	Oui	Oui	Non
Proc EXPORT	Non	Oui	Oui	Non

Pour les fichiers Excel 2007

Nom du moteur	Excel	xls	ExcelCS	PCFILES
LIBNAME	Non	Non	Non	Oui
Proc IMPORT	Non	Non	Oui	Non
Proc EXPORT	Non	Oui	**Oui/ Non**	Non

Pour les fichiers Access (2003 et 2007)

Nom du moteur	Access	AccessCS	PCFILES
LIBNAME	Non	Non	Oui
Proc IMPORT	Non	Oui	Non
Proc EXPORT	Non	Oui	Non

Par expérience, la PROC IMPORT doit être utilisée avec DBMS=XLS plutôt que EXCELCS qui exploite un service spécifique sur Windows et apparaît à l'usage plus lent.

Indépendamment de SAS, il est nécessaire d'utiliser les drivers Office de dernière génération tout en sachant que SAS ne sait pas gérer les toutes dernières versions. Il n'est donc pas vain de suivre les mises à jour et les notes techniques à ce sujet.

Cet article nous a montré que les échanges entre SAS et Excel/Access présentent de sérieux problèmes. Pour y remédier, il est envisageable de créer une base SQL Server qui supporterait la fonction de stockage des données. En effet SAS et Excel/Access communiquent très bien avec ce système de base de données ; SAS utilise un connecteur ODBC pour communiquer avec ce système de base de données.

On pourrait alors imaginer qu'Excel serve d'interface de saisie pour des données qui iraient dans les tables SQL. Ainsi SAS pourrait charger les données par la suite pour

les traitements utilisateurs. Ce genre de projet paraît surdimensionné pour de simples bases mais les utilisateurs rencontreront une meilleure stabilité dans leur Procédure et profiteront des avantages d'une base SQL Server.

6. Liens

Wikipedia : Lien 83.

Support SAS :
Lien 84.
Lien 85.
Lien 86.
Lien 87.
Lien 88.
Lien 89.
Lien 90.
Lien 91.
Lien 92.
Lien 93.

Retrouvez l'article de Stéphane Colas en ligne : Lien 94.

Sortie de PySide 1.1.0

La dernière version en date de PySide, la 1.1.0, vient de sortir. Il s'agit plus d'une version de maintenance que d'une version majeure, après la 1.0.9.

Le seul changement principal concerne le système de types, qui adopte une nouvelle manière de fonctionner, sans oublier les nombreuses corrections de bogues, qui assurent maintenant une bonne compatibilité avec Python 3.2, dont le support a été ajouté à la 1.0.8, de façon assez erratique sur certaines plateformes. Qt 4.8 n'est toujours pas supporté, bien que sorti le mois dernier (Lien 95).

Pour ceux qui désireraient installer cette nouvelle version, les instructions de l'article Installation de PySide : compilation et binaires (Lien 96) devraient toujours être valables pour Python 2 ; pour Python 3, il faut configurer Shiboken d'une manière différente :

```
mkdir build
cd build
cmake .. -DUSE_PYTHON3=ON
make
sudo make install
sudo ldconfig
```

Toutes les informations à jour en anglais sont disponibles sur le Qt Developer Network : Lien 97.

Point important à noter pour cette version : c'est la première à sortir sans le soutien financier de Nokia (Lien 98) (partiellement, cependant, vu qu'il devait s'arrêter fin 2011), cela ne semble pas avoir eu trop d'impact sur le développement, si ce n'est le peu de nouveautés pour cette version - qu'en sera-t-il pour les suivantes ?

Commentez la news de Thibaut Cuvelier en ligne : Lien 99.

Reportage en direct, les Qt Dev Days 2011 à Munich

Lundi 24 octobre 2011

Aujourd'hui, lundi 24 octobre 2011, c'est l'ouverture des Qt Dev Days 2011 à Munich, l'événement de tous les développeurs Qt en Europe.

À 9 h 00 ce matin, des centaines de personnes sont arrivées au Dolce Munich hôtel pour l'inscription et la journée des "trainings" qui va avoir lieu aujourd'hui (cf. photos attachées).

À 10 h 00, cinq salles vont être remplies par des développeurs Qt assistant aux entrainements réalisés par les partenaires de Nokia, à savoir KDAB, basysKom, e-Gits et Digia.



Les sessions d'entrainements sont les suivantes.

- **Bien commencer avec Qt (e-Gits)** : introduction rapide et pratique de Qt basé sur l'utilisation de Qt Core et des QWidgets.
- **Qt avancé, un plongeon en profondeur (KDAB)** : présentation des fondations de Qt les plus importantes à savoir la Vue Graphique (Graphics View), le multithread et la Vue/Modèle (Model/View).
- **Programmation avec Qt Quick (basysKom)** : découverte de la programmation avec Qt Quick et du langage QML.
- **Réaliser des interfaces graphiques avec Qt Quick (KDAB)** : session plus axée sur le côté design des applications Qt Quick.
- **Qt apps avec Nokia - Designer, Développer, Distribuer (Digia, Nokia)** : une mise en bouche du développement de app sur téléphone portable Nokia, du design à la distribution.

En parallèle et à chaque pause, les participants pourront se promener à travers les expositions proposées par Nokia et ses sponsors (cf. photos attachées).


À la suite de ces présentations, une réception organisée par Digia permettra aux développeurs de se rencontrer et également de partager avec l'équipe de Qt (Trolls), impatiente de répondre aux questions des participants à l'événement.

Lars knoll parle de l'Open Gouvernance

Lars Knoll (Chief Architecte Qt)

Évolution de Qt

Comme vous l'avez sûrement découvert, le projet Qt passe en Open Gouvernance (Lien 100). Lars Knoll a apporté des informations sur ce nouveau modèle de développement. Notamment, il décrit les quatre points suivants, au centre du projet :
- l'équitabilité ;
- la transparence ;
- le fait de pouvoir participer ;
- la méritocratie.

Il a même répété plusieurs fois que, si un code n'est pas visible, c'est qu'il n'existe pas au sein du projet (principe de transparence).

Les bénéfices présentés sont :
- une stabilité accrue (car plus de développeurs peuvent corriger le code et le tester) ;
- des versions plus riches (plus de gens pourront apporter du contenu) ;
- une visibilité accrue ;
- que Qt corresponde plus à vos besoins (et, s'il ne le fait pas, vous pouvez toujours proposer / développer vos morceaux de code) ;
- de meilleures opportunités pour les experts Qt.

Le système fonctionne suivant la hiérarchie suivante (hiérarchie pyramidale):

- tout en haut, on retrouve le mainteneur en chef (au moins un par module de Qt) ;
- les mainteneurs ;
- les approbateurs (qui regardent le code soumis et qui l'acceptent) ;
- les contributeurs (rapport de bogues / patches).

Finalement, M. Knoll a donné les premiers chiffres de ce système. Entre vendredi et aujourd'hui (mardi), il y a eu :
- 270 nouveaux comptes ;
- 53 contributions ;
- 33 patches acceptés.

Cela prouve que le système est déjà en marche.

Le futur de Qt

Tout d'abord, parlons un peu de Qt 4.8, qui va bientôt être disponible. Qt 4.8 apporte :
- Qt Quick 1.1 ;
- meilleures performances ;
- support de HTML5 dans QWebkit ;
- facilité du portage sur de nouvelles plateformes.

Rapidement, M. Knoll a abordé le sujet de Qt 5. Avec Qt 5, une nouvelle structure du code a été mise en place. De plus, cette version de Qt 5 s'oriente sur le développement d'applications utilisant des interfaces utilisateur fluides et répondant au toucher.

Mais ce n'est pas parce que cette orientation existe que les développeurs bureau sont abandonnés. En effet, les QWidget sont toujours d'actualité. Ainsi, comme on a déjà pu le voir avec les dernières versions de Qt, deux façons sont possibles pour développer une application :
- la voie habituelle en C++ ;
- la voie nouvelle en QML.

Lars Knoll a aussi décrit les nouveautés / changements dans Qt 5 :
- Qt 5 sera modulaire ;
- un état de l'art des interfaces utilisateur (toujours plus fluides / plus animées) ;
- fonctionne entièrement dans l'écosystème ;
- **compatible Qt 4.x** ;
- Qt Quick 2.0 ;
- une toute nouvelle pile graphique ;
- une nouvelle structure ;
- un meilleur support des plateformes.

En bref :
- dans Qt 5, on ne parlera plus de Qt Mobility ;
- les widgets sont vus comme add-ons (car les applications mobiles peuvent ne pas en avoir besoin).

Pour le rendu, un nouveau paradigme a été mis en place, il est possible d'utiliser un rasteriser logiciel, en place avec le système QPainter, ou alors d'utiliser Qt Quick 2.0 entièrement construit et optimisé pour OpenGL ES.

Il a été dit que même le rasteriser logiciel obtient de bonnes performances.
À propos de cela et en lien avec l'Open Gouvernance, le groupe DirectFB a déjà publié du code sur le projet Qt.

Le portage d'application Qt 4 sera facile et composé des points suivants :
- exécution d'un script pour les en-têtes ;
- suppression de `#include <QMimeData>` ;
- peu d'incompatibilités.

Il a insisté sur le fait que les applications Qt 4 seront compatibles (à part quelques exceptions) avec Qt 5. Nokia cherche à éviter les problèmes qu'ils ont eus auparavant, lors du passage de Qt 3 à Qt 4.

De plus, M. Knoll a annoncé que Nokia ferait mettre à disposition deux versions de Qt par an, suivant la rapidité des versions que l'on retrouve chez Ubuntu ou encore Firefox.

Le planning actuel pour Qt 5 est :
- arrêt d'intégration des fonctionnalités, début 2012 (rien de nouveau n'est ajouté, il n'y aura que de la résolution de bogues à partir de cette date) ;
- version bêta durant le printemps 2012 (mars / avril) ;
- version finale dans la première moitié de l'année 2012.

Pour finir, Lars Knoll a décrit les possibles développements futurs pour Qt :
- un support complet de Qt Quick pour les plateformes de bureau ;
- une intégration d'un rendu logiciel OpenGL (avec MESA : Lien 101) ;
- QWebkit 2 ;

- un support du JavaScript étendu sur plus d'architecture de processeurs.

Lundi 24 octobre 2011

Nous voici aujourd'hui à la seconde journée de la huitième édition des Qt Dev Days 2011 à Munich, avec une matinée rythmée par les traditionnelles "keynotes".

La première présentation réalisée par Marco Argenti (SVP Nokia Developer Experience) a donné une idée de la stratégie de Nokia autour de Qt. Aujourd'hui, 100 millions de smartphones actifs avec Qt installé sont sur le marché, cela représente un nombre de téléchargements de 9 millions d'applications par jour. Le N9, récemment sorti, est de loin le meilleur téléphone jamais réalisé pour les développeurs. Qt est devenu très populaire au cours des dernières années et reste un investissement stratégique pour Nokia en concordance avec le Qt Project. Nokia utilise d'ailleurs Qt pour ses propres outils internes notamment le logiciel Nokia Store.

Pour plus d'information sur la keynote, retrouvez les commentaires de Vincent Meyer sur place : Lien 102.

La seconde keynote présentée par Rick Spencer (Director of Engineering chez Canonical) a permis d'avoir une vue sur un autre projet, Open Source depuis quelques années, à savoir : Ubuntu. Il s'est agi principalement d'une présentation de Ubuntu, de son fonctionnement Open Source et d'informations sur l'utilisation de Qt par Ubuntu. Le nouveau système de fenêtrage fourni avec Ubuntu par défaut utilise Qt Quick (QML) dans sa version 2D pour netbooks. Cela a l'avantage de fournir par défaut toutes les bibliothèques nécessaires au bon fonctionnement d'applications Qt. Ubuntu est probablement la distribution Linux la plus utilisée avec un nombre d'utilisateurs approximatif de 20 millions.

Pour plus d'informations sur la keynote, retrouvez les commentaires de Vincent Meyer sur place : Lien 102.

Une troisième session par Louis Gump (Vice-Président CNN Mobile) sur l'application CNN pour téléphone portable Nokia était très enrichissante. La discussion était découpée en trois parties :

- comment les téléphones portables changent notre société aujourd'hui ;
- comment les téléphones portables changent CNN ;
- comment les innovations nous aident à suivre le rythme.

Pour terminer cette matinée, Lars Knoll (Chief Maintener du Qt Project) a présenté Qt 4.8, Qt 5 et l'Open Gourvenance via Qt Project démarré vendredi dernier (21 octobre 2011), la partie probablement la plus attendue par l'ensemble des développeurs présents pour savoir où va Qt dans l'avenir proche comme dans l'année future.

Le Qt Project étant maintenant en ligne (Lien 103), les développeurs Qt (communauté et commercial) peuvent participer directement à Qt de différentes façons (code source, rapport de bogues, tests, etc.). Tout l'écosystème de Qt fait partie du projet à savoir, les outils, Qt 4 et Qt 5, etc. Plus de personnes contribuant à Qt permet d'avoir au final plus de testeurs pour l'ensemble du code ce qui va améliorer la qualité du Framework dans son ensemble.

Qt 4.8 va sortir d'ici la fin de l'année avec Qt Quick 1.1, de meilleures performances, des nouvelles fonctionnalités HTML5 et un portage plus facile sur différentes plateformes.

Qt 5 sera disponible dans la première moitié de 2012 avec une compatibilité Qt 4.x malgré la restructuration modulaire. Qt Quick 2.0 avec OpenGL ES fera partie de Qt 5 ainsi que l'intégration complète de Qt Mobility. Une séparation nette entre Qt Essentials et Qt Add-ons sera cependant faite vous permettant d'emporter avec votre application seulement ce dont vous avez besoin.

Pour plus d'information sur la keynote, retrouvez les commentaires de Alexandre Laurent sur place : Lien 104.

Une interview avec Lars Knoll à la suite de sa keynote a permis d'obtenir les informations suivantes :

- il n'y aura pas de compatibilité binaire mais une compatibilité du code source entre les deux branches majeures, Qt 4.x et Qt 5, avec quelques exceptions mineures ;
- Qt Network sera un module séparé et non inclus dans Qt Core, comme suggéré au Contributor Summit en juin dernier ;

- la structure de l'*Open Gourvernance*, composée de Maintainers et Approvers, sera pour commencer principalement composée de personnes de Nokia avec quelques personnes d'autres entreprises telles que Thiago Macieira (Intel) ;
- les deadlines seront respectées dans le sens où, si une fonctionnalité n'est pas prête pour une date précise, au lieu de retarder tout le monde en attendant qu'elle soit prête, elle sera tout simplement repoussée à une version mineure ultérieure, permettant de respecter le cycle de deux nouvelles versions par an ;
- la structure actuelle de release à savoir Technical Preview, Beta, Release Candidate, etc. sera utilisée pour le moment mais, en cas de nécessité éventuelle, changée pour une plus adaptée.

Commentez la news de Jonathan Courtois et Alexandre Laurent en ligne : Lien 105.

Les dernier tutoriels et articles

Le livre blanc Qt (suite de la parution du numéro précédent)

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.

6. Graphique et multimédia

Qt fournit un excellent support pour les graphismes en 2D et 3D. Les classes 2D de Qt supportent les graphismes vectoriel et matriciel, peuvent charger et enregistrer une large gamme extensible de formats d'image et exporter du texte et des graphismes dans des fichiers PDF. Qt peut dessiner du texte riche Unicode transformé, des documents SVG (Scalable Vector Graphics) et fournit un canevas pleinement fonctionnel pour des applications interactives plus poussées. Qt fournit aussi des fonctionnalités pour lire des fichiers et des flux audio et vidéo.

Les graphismes sont affichés en utilisant des objets de dessin indépendants du matériel, qui permettent au développeur de réutiliser le même code sur différents types de périphériques, représentés dans Qt par des périphériques de dessin. Cette approche assure qu'une large gamme de puissantes opérations de dessin sont disponibles pour chaque plateforme supportée par Qt et permet également aux développeurs de choisir les périphériques les plus adaptés à leurs besoins.

La démonstration des boîtes présente une série de fonctionnalités graphiques de Qt.

Les applications graphiques qui requièrent un canevas interactif peuvent tirer profit du framework de la vue graphique (*graphics view*) pour gérer et rendre les scènes avec un grand nombre d'éléments interactifs, en utilisant de multiples vues si nécessaire.

Le support de Qt pour OpenGL et OpenGL ES aide les développeurs à intégrer des graphismes 3D dans leurs applications, en même temps qu'il leur permet de profiter du matériel graphique moderne pour améliorer les performances de rendu 2D.

Le support des couleurs de manière indépendante du matériel permet de spécifier des couleurs par des valeurs RVBA, HSVA ou CMJNA ou par des noms usuels. Les canaux de couleur utilisés font seize bits de largeur et un niveau d'opacité optionnel peut être spécifié. Qt alloue automatiquement la couleur demandée dans la palette système ou utilise une couleur similaire sur des affichages limités.

6.1. Dessin

Qt fournit une API indépendante de la plateforme pour dessiner sur des widgets et d'autres périphériques de dessin. Il fournit des primitives de dessin comme des fonctionnalités avancées (notamment, les transformations). Tous les widgets de Qt se dessinent à l'aide de QPainter et les programmeurs utilisent cette même classe pour implémenter leurs widgets personnalisés.

QPainter fournit des fonctions standard pour dessiner des points, des lignes, des ellipses, des arcs, des courbes de Bézier et d'autres primitives. Des opérations de dessin plus complexes incluent le support des polygones et des chemins vectoriels, permettant la préparation des dessins détaillés (dessinables alors avec un simple appel de fonction). Le texte peut aussi être affiché directement avec un dessinateur ou incorporé dans un chemin pour

utilisation future.

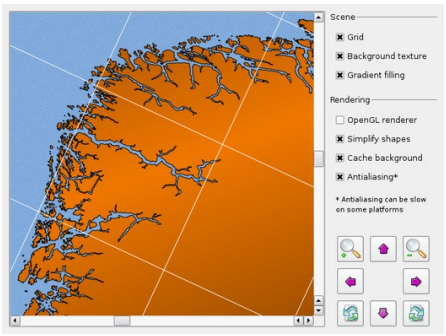
Le système de dessin de Qt fournit aussi un certain nombre de fonctionnalités pour améliorer la qualité globale du rendu, dont des fondus alpha, des modes de composition Porter-Duff, l'anti-aliasing, des remplissages avec des dégradés linéaires, radiaux et coniques.

6.2. Images

Qt supporte l'entrée, la sortie et la manipulation d'images dans divers formats, dont BMP, GIF, JPEG, MNG, PNG, PNM, TIFF, XBM et XPM. Les deux classes (QImage et QPixmap) peuvent être utilisées comme périphériques de dessin et utilisées dans des applications graphiques interactives ou pour préparer les images pour utilisation future dans des composants de l'interface.

QImage est utilisée pour la manipulation d'images et peut effectuer des conversions entre diverses profondeurs de couleur et formats de pixel. Les programmeurs peuvent manipuler les pixels et les données de la palette, appliquer des transformations comme des rotations et réduire la profondeur de couleur avec tramage si désiré. Le support des canaux alpha permet d'utiliser la transparence et les fusions alpha pour la composition d'images et à d'autres fins.

La gamme de formats de fichiers supportés qui peuvent être utilisés avec ces classes peut être étendue avec des plug-ins d'extension.



Le framework de la vue graphique permet la création d'applications graphiques qui combinent une haute qualité de rendu à des fonctionnalités complètes pour l'interaction avec l'utilisateur.

6.3. Périphériques de dessin et impression

QPainter peut être utilisé sur n'importe quel périphérique de dessin. Le code requis pour dessiner sur n'importe quel périphérique supporté est le même, peu importe le périphérique.

Tous les widgets sont des périphériques de dessin. Les fenêtres transparentes avec une forme spécifique peuvent être créées sur des systèmes correctement configurés.

Les surfaces OpenGL utilisées avec QGLWidget sont aussi des périphériques de dessin qui convertissent les appels standard à QPainter en appels OpenGL, ce qui fait que des opérations 2D sont accélérées sur des périphériques avec un matériel supporté.

Des images peuvent aussi être créées en dessinant sur des QImage (indépendants du matériel) et des objets QPixmap, optimisés pour l'affichage. Les formats de fichier standard peuvent être créés en effectuant un rendu dans une image avec la profondeur de couleur et le format de pixels désirés. Les images peuvent être créées avec support de différents niveaux de transparence et dessinées sur des widgets personnalisés pour obtenir certains effets.

Les fichiers vectoriels et les métafichiers sont aussi supportés par le système de dessin. QSvgGenerator crée des images SVG en traduisant les commandes de dessin par les structures correspondantes du format SVG. QPicture est utilisé pour contenir une série de commandes de dessin qui peuvent être rejouées pour dessiner sur un autre périphérique ou pour stockage dans un fichier.

L'impression est effectuée avec un rendu dans un QPrinter, qui représente une imprimante physique. Sur Windows, les commandes de dessin sont envoyées au moteur d'impression de Windows, qui utilise les pilotes installés. Sur UNIX, les données PostScript ou PDF sont envoyées au démon d'impression - ceci est géré par CUPS (Common Unix Printing System) sur des systèmes récents.

En utilisant l'API de dessin générique de Qt, des applications peuvent créer des fichiers PDF et PS qui peuvent être générés sur toutes les plateformes, donc créer des documents de haute qualité qui peuvent être visualisés par des applications de lecture appropriées.

6.4. Framework de la vue graphique

Qt contient un nouveau mais très puissant framework pour les applications graphiques interactives, utilisé pour gérer et afficher un grand nombre d'items dans une scène 2D. La vue graphique fournit une API orientée objet pour ajouter de nouveaux items à une scène et une API plus traditionnelle, semblable à celle du canevas, contenant des fonctions auxiliaires pour créer des items prédéfinis.

Une fois créés, les items peuvent être placés avec les position, orientation et échelle requises dans une scène. Les fonctionnalités d'affichage et de gestion des items sont implémentées séparément dans les classes QGraphicsView et QGraphicsScene, ce qui facilite des fonctionnalités comme les multiples vues sur la même scène et le support de moteurs de rendu échangeables.

Une sélection de types d'item standard est fournie, ceux-ci peuvent être étendus par le biais de sous-classes pour fournir des types d'item personnalisés. Ils peuvent être groupés pour un contrôle de plus haut niveau de la scène. Chaque scène, vue et item fournit un ensemble complet de fonctions pour transformer les coordonnées entre les différents systèmes. Les items standard et personnalisés peuvent être rendus sélectionnables et déplaçables, avec un niveau basique d'interactivité et un minimum de code.

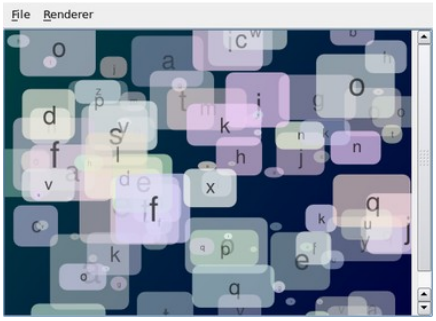
La vue graphique a été pensée avec les animations en tête : les items peuvent être utilisés pour créer des objets animés qui sont transformés en fonction d'une série de transformations définies à certains points sur une ligne du temps.

Certains items standard apportent des fonctionnalités

trouvables à d'autres endroits de Qt au framework, dont des éditeurs de texte riche, la navigation sur le Web, l'affichage de dessins SVG et l'utilisation de formes, chemins et images. Les items d'une scène peuvent être rendus indépendamment de la vue attachée, les scènes peuvent donc être rendues dans des images et imprimées dans des fichiers PDF.

Un modèle événementiel complet permet une gestion efficace des événements en les distribuant aux items qui en ont besoin. Puisque la gestion basique des items est effectuée par le framework, les items doivent seulement répondre aux événements s'ils ont besoin d'une information particulière sur leur environnement.

Les applications qui doivent mélanger des éléments d'interface utilisateur classique avec du contenu interactif peuvent embarquer des widgets directement dans une scène en utilisant QGraphicsProxyWidget ou créer – en partant de rien avec QGraphicsWidget – des éléments ressemblant à des items. Comme pour les interfaces conventionnelles, les gestionnaires de disposition peuvent être utilisés pour disposer les widgets et les items dans une scène.



Les dessins SVG peuvent être rendus sur n'importe quel périphérique de dessin supporté par Qt.

6.5. SVG

Scalable Vector Graphics est un format de fichier basé sur XML et un langage pour décrire des applications graphiques, habituellement associé aux graphismes 2D du Web. Le support SVG dans Qt est basé sur le standard SVG 1.1, une recommandation du W3C (World Wide Web Consortium). Il fournit des fonctionnalités additionnelles pour le support des profils Tiny de SVG 1.1 et 1.2.

Qt peut rendre des dessins SVG sur n'importe quel périphérique de dessin, dont ceux pour les images et les widgets OpenGL. Cette flexibilité laisse les développeurs échanger la qualité pour la rapidité quand cela est nécessaire. Les dessins SVG peuvent aussi être utilisés pour des icônes dans des contrôles d'interface, enlevant ainsi le besoin de générer des bitmaps dans une série de tailles prédéfinies.

Les développeurs peuvent aussi générer des dessins SVG en utilisant les fonctions de QPainter pour dessiner sur un périphérique de dessin spécialisé SVG, les graphiques utilisés dans des applications peuvent donc être exportés comme dessins SVG avec peu d'effort supplémentaire.

6.6. Graphismes 3D

OpenGL est une API standard pour le rendu de graphismes 3D qui peut être utilisée par les développeurs Qt pour inclure des graphismes 3D dans leurs applications graphiques. Le module OpenGL de Qt est disponible sur Windows, X11 et Mac OS X et utilise la bibliothèque OpenGL de chaque système.

Pour utiliser OpenGL dans une application Qt, il suffit aux développeurs de sous-classer QGLWidget et de dessiner dessus avec les fonctions standard d'OpenGL. Qt fournit des fonctions pour convertir les valeurs des couleurs dans le format d'OpenGL pour fournir une interface constante pour les applications.

Qt supporte aussi des fonctionnalités et extensions d'OpenGL, avec une utilisation plus pratique à l'intérieur des applications Qt. Des fonctions auxiliaires permettent de créer des textures depuis des images et le support des tampons de pixels et de frames est fourni par des classes appropriées. Le support de fonctionnalités comme les tampons d'échantillons peut être activé si elles sont disponibles sur le système sous-jacent.

Les applications 2D peuvent utiliser des sous-classes de QGLWidget pour améliorer les performances de rendu sur du matériel approprié. Dans ce cas, les opérations standard de QPainter sont transformées en appels OpenGLCeci rendu aussi possible de recouvrir des contrôles sur des scènes 3D peintes avec seulement OpenGL. Sur des plateformes embarquées, où l'accélération matérielle est souvent limitée, ce moteur de rendu est restreint aux fonctionnalités d'OpenGL ES 2.0, ce qui permet de s'assurer qu'il fonctionne parfaitement sur un grand nombre de périphériques.

Sur du matériel approprié, le support du rendu anti-aliasé peut être activé pour améliorer la rapidité du rendu et la qualité des graphismes produits en utilisant le moteur OpenGL. Sur du matériel moins puissant, les développeurs peuvent donner le choix à l'utilisateur entre la qualité et la rapidité, en exposant ces options de rendu à l'utilisateur à l'exécution.

6.7. Multimédia

Qt utilise le framework multimédia Phonon, un projet open source de KDE, pour fournir des fonctionnalités de lecture de médias, à travers une API constante et multiplateforme. Qt s'assure que les applications sur Linux, Unix, Windows et Mac OS X utilisent de manière transparente le framework multimédia de chaque plateforme - ceci signifie que les applications peuvent aussi profiter du support spécifique à la plateforme pour les codecs et formats audio et vidéo.

Les fonctionnalités de Phonon peuvent être intégrées à d'autres technologies de Qt. Par exemple, des widgets de films peuvent être ajoutés à des pages Web affichées avec le moteur d'affichage WebKit et à des scènes rendues avec le framework de la vue graphique.

Des classes supplémentaires pour le multimédia sont incluses dans le module QtMultimedia. Ces classes sont orientées sur l'accès de bas niveau aux données audio et vidéo.

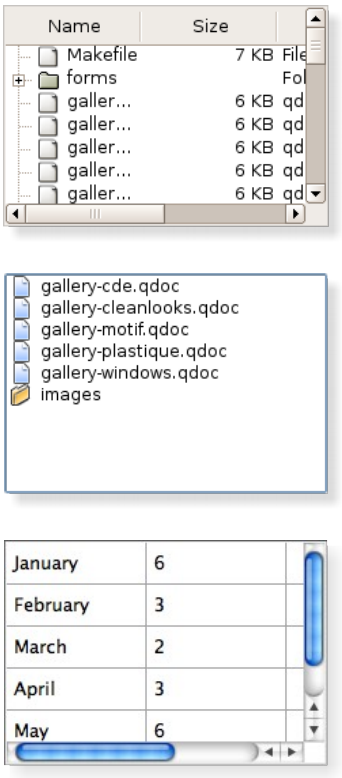
6.8. Références en ligne

- Qpainter : Lien 106.
- La vue graphique : Lien 107.
- QtOpenGL : Lien 108.
- QtMultimedia : Lien 109.
- QtSVG : Lien 110.
- Phonon : Lien 111.

7. Vues d'items

Les widgets de visualisation d'items fournissent des contrôles GUI standard pour afficher et modifier de larges quantités de données. Le framework modèle-vue sous-jacent isole le stockage des données et leur présentation à l'utilisateur, ce qui fait que des fonctionnalités comme le partage de données, le tri et le filtrage, les vues multiples et les multiples représentations peuvent utiliser le même ensemble de données.

Lors de l'écriture d'applications qui traitent de grandes quantités de données, les développeurs utilisent généralement des widgets de vue d'items pour afficher les données rapidement et efficacement. Des vues standard que l'on peut trouver dans des boîtes à outils graphiques modernes incluent généralement des vues en liste contenant de simples listes d'items, des vues en arbre avec une liste hiérarchique d'items et des tableaux, qui fournissent des fonctionnalités de positionnement semblables aux tableurs.



Qt fournit des vues standard pour les arbres, listes et tableaux d'items.

Les classes de vue d'items de Qt sont disponibles en deux formats : des widgets classiques et des composants modèle-vue. Les listes, tableaux et arbres classiques sont des widgets autonomes qui gèrent leurs objets d'item, explicitement créés par le développeur.

QListView, QTableView et QTreeView ont les composants modèle-vue équivalents. Ils fournissent une manière plus propre et plus orientée composant de gérer les ensembles de données. De plus, une série de modèles standard de données sont fournis pour aider les développeurs à organiser leurs données.

7.1. Vues d'items standard

Les implémentations standard des widgets de vue en liste, en arbre, en tableau et d'icônes sont fournies par Qt. Elles supportent les opérations de glisser-déposer dans la même vue et entre des vues différentes. Comme pour tous les widgets Qt, elles sont aussi complètement intégrées au système de ressources de Qt.

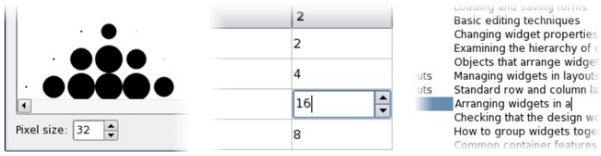
Les classes de vue sont utilisées pour afficher des données de diverses boîtes de dialogue dans Qt et sont utilisées de manière extensive dans Qt Designer, Qt Assistant et Qt Linguist.

Les vues classiques servent généralement à afficher et gérer quelques centaines d'items, en usant d'une architecture qui utilise des objets individuels pour encapsuler des données. Cette approche devrait être familière à des développeurs Qt et fournit une manière agréable de construire rapidement des interfaces riches pour gérer des quantités raisonnables de données.

Pour la cohérence et la fiabilité, les vues classiques se basent sur le framework modèle-vue de Qt, qui fournit une manière plus adaptée à de grands volumes de données et plus personnalisable pour gérer ces vues.

7.2. Le framework modèle-vue

Le framework modèle-vue fourni par Qt est une variation du patron de conception MVC bien connu (*Model-View-Controller*, modèle-vue-contrôleur), spécifiquement adapté aux vues d'items de Qt. Dans cette approche, les modèles sont utilisés pour fournir des données à d'autres composants, des vues qui affichent des items à l'utilisateur et des délégués qui gèrent le rendu et l'édition.



L'architecture orientée composant du framework modèle-vue facilite la personnalisation des vues d'items.

Les modèles ne sont que de fins emballages autour des sources de données, écrites conformément à une interface standard fournie par QAbstractItemModel. Cette interface autorise les widgets dérivant de QAbstractItemView à accéder aux données fournies par le modèle, peu importe la nature de la source des données.

La séparation entre les données et leur représentation dans cette approche fournit un certain nombre d'avantages par rapport aux vues classiques :

- puisque les modèles fournissent une interface standard pour accéder aux données, ils peuvent être conçus et écrits séparément, même remplacés au besoin ;

- les données obtenues de modèles peuvent être partagées par les vues ; ainsi, des applications peuvent offrir plusieurs vues des mêmes données, potentiellement avec plusieurs représentations ;
- les sélections peuvent être partagées entre les vues ou gardées séparées, en fonction des attentes et demandes de l'utilisateur ;
- pour les listes, arbres et tableaux standard, le gros du rendu est effectué par les délégués, ce qui facilite la personnalisation des vues sans devoir réécrire beaucoup de code ;
- en utilisant des *modèles proxy*, les données fournies par des modèles peuvent être transformées avant d'être passées aux vues ; les applications peuvent donc fournir des fonctionnalités de tri et de filtrage que l'on peut partager entre les vues.

Le système modèle-vue est aussi utilisé par les modèles SQL de Qt pour rendre l'intégration d'une base de données plus simple pour les développeurs ne les maîtrisant pas.

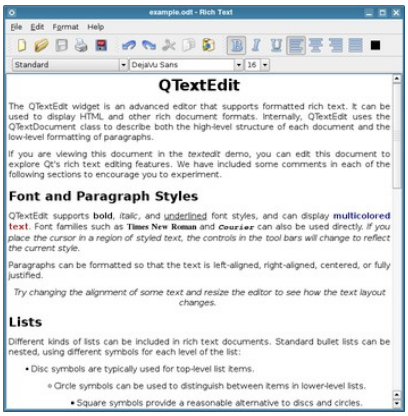
7.3. Références en ligne

- Programmation modèle-vue : Lien 112.
- Exemples : Lien 113.

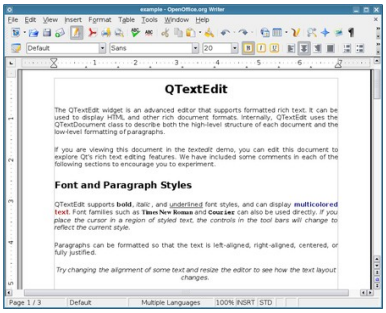
8. Gestion du texte

Qt fournit un puissant éditeur de texte dans lequel l'utilisateur peut créer et éditer des documents riches, il peut aussi être utilisé pour préparer des documents pour l'impression. La structure de document sous-jacente est complètement accessible aux développeurs, ce qui fait que la structure et le contenu du document peuvent être manipulés.

Les documents riches contiennent généralement du texte dans plusieurs fontes (police, taille, corps) et couleurs, arrangé dans une série de paragraphes. Il peut aussi être organisé dans des listes et des tableaux, ainsi qu'être visuellement séparé du corps du document avec des cadres. L'apparence de chaque élément du document peut être précisément ajustée en utilisant les nombreuses fonctionnalités disponibles aux développeurs grâce à l'API de texte riche.



Les fonctionnalités avancées d'édition de texte riche de Qt permettent la création et l'édition de documents complexes dans QTextEdit.



Les documents peuvent être exportés au format OpenOffice pour utilisation dans un logiciel de traitement de texte.

8.1. Édition de texte riche

L'affichage interactif de texte riche et son édition sont gérés par Qt dans les widgets QTextBrowser et QTextEdit. Ils supportent complètement Unicode et sont construits sur une représentation de la structure du document fournie par QTextDocument, il n'y a pas besoin d'utiliser des langages de balisage intermédiaires pour créer du texte riche. QTextDocument fournit aussi un support de l'import et de l'export d'un sous-ensemble de HTML 4.0, des capacités d'undo-redo (dont les actions groupées) et la gestion des ressources.

Qt fournit une API orientée objet pour les documents qui aide le développeur à obtenir une vue globale de haut niveau sur les structures. Une API basée sur des curseurs est aussi fournie pour faciliter l'exploration, le traitement et la transformation des documents. En plus des classes correspondant à la structure et au contenu, il y a bon nombre d'autres classes pour contrôler l'apparence du texte et des éléments de document. Ceci fait que les styles de texte pour les tableaux, les listes, les cadres et les paragraphes ordinaires peuvent être personnalisés pour donner aux documents l'apparence désirée.

Les documents créés par le code restent éditables dans les widgets QTextEdit et conservent un historique complet d'undo-redo. Les développeurs peuvent ajouter de nouvelles fonctionnalités à l'éditeur, faisant en sorte que les utilisateurs puissent alors insérer des structures et du contenu personnalisés.

8.2. Personnalisation, impression et export du document

Les fonctionnalités de gestion du texte de Qt peuvent être utilisées pour fournir du formatage spécialisé du texte pour des widgets personnalisés et des documents riches. On peut les écrire avec des classes de bas niveau comme QTextLayout pour disposer le texte ligne par ligne, puis les intégrer dans le système de disposition extensible de texte fourni par QTextDocument pour utilisation avec QTextEdit.

Des règles de coloration syntaxique peuvent aussi être appliquées à des documents riches avec QSyntaxHighlighter. Un simple widget QTextEdit pourra être utilisé comme base d'un éditeur de code, cela peut aussi servir à des outils de recherche dans le document.

Les documents peuvent aussi être formatés selon les informations obtenues d'une boîte de dialogue

QPrintDialog dans une série de pages que l'on peut passer à un QPrinter.

La classe QTextDocumentWriter fournit un support de l'export de documents en HTML, texte brut et fichiers *OpenDocument Format* (ODF). Cette classe expose ses fonctionnalités par le biais d'une API générique et est prévue pour être étendue à d'autres formats dans de futures versions.

8.3. Références en ligne

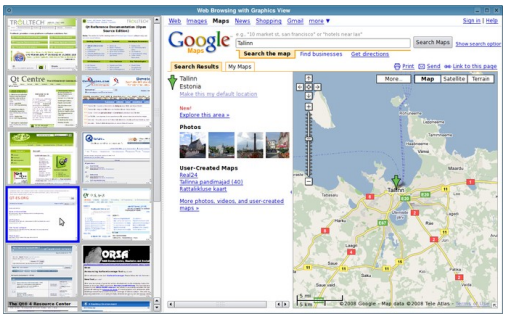
- Les classes du framework Scribe : Lien 114.
- Traitement du texte riche : Lien 115.
- QtextDocumentWriter : Lien 116.

9. Intégration Web avec QtWebKit

L'intégration de Qt avec le moteur WebKit fait que les développeurs peuvent introduire des fonctionnalités Web dans leurs applications avec des API et paradigmes de la même veine que Qt pour afficher et interagir avec du contenu Web.

Qt inclut un support intégré pour Web Kit, un moteur de rendu Web open source complet, qui se focalise sur la stabilité et les performances. La version de WebKit fournie avec Qt supporte bon nombre de standards du Web, dont HTML 4.01, XHTML 1.1, CSS 2.1 et JavaScript 1.5. Des fonctionnalités plus avancées sont aussi disponibles, elles sont présentées dans le livre blanc dédié.

La gestion du réseau de WebKit est assurée de manière transparente par les classes de Qt, qui donnent aux composants du navigateur une implémentation conforme aux standards HTTP 1.1, supportant la communication SSL (*Secure Sockets Layer*) et les proxies.



Les widgets Qt peuvent être embarqués dans des pages Web affichées par l'intégration WebKit dans Qt ; ces pages Web peuvent être affichées comme des items avec le framework de la vue graphique.

9.1. Intégration aux applications natives

Le support de Qt pour WebKit dépasse l'affichage de HTML en exposant les fonctionnalités de WebKit aux applications en utilisant les paradigmes Qt. Par exemple, le mécanisme de signaux et de slots pour la communication facilite la connexion de composants Web à des widgets et à d'autres objets de l'application.

Inversement, l'intégration entre Qt et le moteur de rendu permet d'intégrer des contrôles Qt natifs dans les pages Web, ce qui rend possible de combiner du contenu Web avec des interfaces natives hautement dynamiques.

Les applications peuvent aussi profiter de WebKit pour utiliser le stockage natif pour les données persistantes, cette fonctionnalité étant supportée par Qt. Les développeurs peuvent activer le stockage natif pour des applications qui interagissent avec des services distants et prendre appui sur les options de configuration pour définir un endroit approprié et un quota sur le système utilisateur.

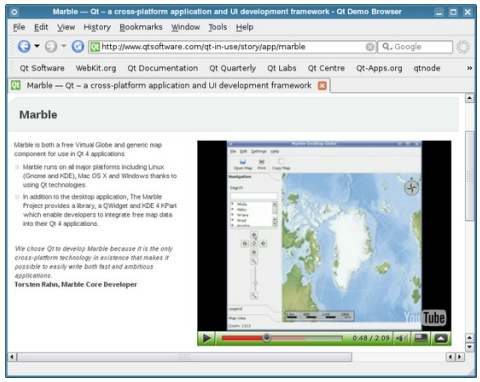
9.2. API d'accès à l'arbre DOM

La manière standard de manipuler la structure de pages Web est d'utiliser l'API DOM (*Document Object Model*, modèle objet du document). QtWebKit inclut une implémentation de l'API des sélecteurs du W3C qui fournit un accès très simple à la structure des pages, ainsi que leur modification.

Cette API rend intuitif l'accès au DOM en laissant les développeurs réutiliser leurs connaissances sur les sélecteurs CSS, sans occasionner de surcharge ou de maintenance supplémentaire.

9.3. Support des plug-ins Netscape

Les plug-ins se conformant à l'API Netscape, un standard *de facto* pour les composants tiers des navigateurs, peuvent être embarqués et affichés dans des pages Web rendues par QtWebKit. La configuration de cette fonctionnalité est effectuée dans une classe Qt qui est aussi utilisée pour configurer d'autres sortes de plug-ins, comme les plug-ins de widgets exposés par l'application à l'environnement Web.



Les plug-ins tiers sont supportés par QtWebKit via l'API Netscape.

9.4. Références en ligne

- Le module QtWebKit : Lien 117.
- Démo : un navigateur Web : Lien 118.
- Intégrer des widgets avec QtWebKit : Lien 119.

Retrouvez l'intégralité de l'article des Qt Developer Network traduit par Aldiemus et Thibaut Cuvelier en ligne : Lien 120

Utilisation de QGraphicsItem

Ce tutoriel traite de l'utilisation des QGraphicsItem. Il est destiné aux développeurs Python ayant déjà de bonnes connaissances dans l'usage du framework Qt.

1. Introduction

QGraphicsItem est la classe de base pour tous les objets graphiques pouvant être affichés dans un **QGraphicsScene**. Cette classe permet de créer ses propres objets graphiques et de gérer leurs propriétés telles que les données de géométrie de l'objet et ses interactions avec les autres éléments présents dans la scène (dont la détection de collisions entre éléments graphiques).

2. Les items graphiques de base

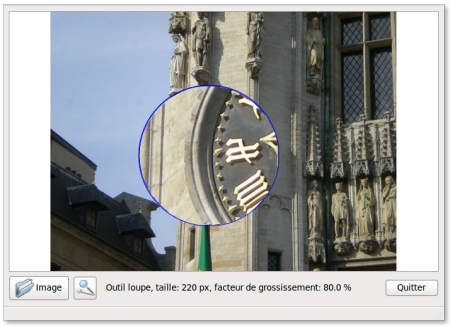
Qt propose une série de **QGraphicsItem** de base destinés à représenter les objets graphiques les plus courants :

- **QGraphicsEllipseItem** : définit un item ellipse ;
- **QGraphicsLineItem** : définit un item ligne ;
- **QGraphicsPathItem** : définit un item chemin de tracé ;
- **QGraphicsPixmapItem** : définit un item pixmap ;
- **QGraphicsPolygonItem** : définit un item polygone ;
- **QGraphicsRectItem** : définit un item rectangle ;
- **QGraphicsSimpleTextItem** : définit un item simple texte ;
- **QGraphicsTextItem** : définit un item texte plus complexe que l'item précédent.

Nous nous intéresserons plus particulièrement, dans cet article au **QGraphicsPixmapItem** en réalisant un objet graphique utilisant à la fois les classes **QPixmap**, **QPainter**, **QPainterPath**, l'application de texture, la découpe de zone transparente, les évènements souris et clavier et les signaux pour communiquer avec l'interface utilisateur.
Pour cela nous réaliserons un outil loupe pour visionneuse d'image.

3. Les scripts Python

En bas de cet article, se trouvent les liens permettant de télécharger l'archive contenant les scripts nécessaires. Le code ***main.py*** est celui qu'il faut lancer, ***magnifier.py*** est celui de la loupe et sera principalement décrit ici et le script ***miniView.py*** est une version simplifiée de la visionneuse décrite dans cet article : Lien 121, et dont le code ne demande plus à être détaillé.



4. Principes de base

Voyons d'abord le fonctionnement de la loupe. On part du principe que l'on dispose d'une image créée en tant que **QImage** à sa taille réelle et une pixmap **QPixmap** issue de l'image, réduite aux dimensions de la vue et affichée dans celle-ci.

La loupe aura trois paramètres : sa forme, sa taille et son facteur de grossissement.

L'image créée en premier étant la référence nécessaire de la visionneuse pour recréer la pixmap lors de zoom, on va en créer une copie et la mettre directement au facteur de grossissement choisi pour la loupe. On déposera ensuite sur cette copie un gabarit de la forme et de la taille de la loupe, qui servira à prélever la texture qui sera « peinte » dans le **QGraphicsPixmapItem**. Ensuite, par un système de pantographe informatique on assurera le déplacement du gabarit sur l'image de référence suivant les déplacements de la loupe effectués par l'utilisateur.

La classe **Magnifier** se présentera comme ceci :

```
magnifier.py

class Magnifier(QtGui.QGraphicsPixmapItem):
    def __init__(self, main, parent=None,
graphic=None, scene=None,
                                                i
mg=None, zoom=None):
        super(Magnifier, self).__init__(parent,
scene)
        self.main = main
        d = dict(shape = "square",
                 size = 160,
                 factor = 1.0,
                 image = img,
                 graphic = graphic,
                 scene = scene,
                 cur_pos = None,
                 zoom = zoom,
                 background_color = (255, 255,
255, 255),
                 old_X = None,
                 old_Y = None)
        for key, value in d.iteritems():
            setattr(self, key, value)

        self.setFlags(QtGui.QGraphicsItem.ItemIsM
ovable |
                      QtGui.QGraphicsItem.ItemI
sFocusable)
        self.setAcceptHoverEvents(True)

        self.build_settings()
        self._moved = Moved()
        self._optionChanged = OptionChanged()
```

Les arguments nécessaires à l'instanciation de la classe sont les suivants :

- ***main*** : l'application principale ;
- ***parent*** : le parent, généralement laissé à None ;
- ***graphic*** : l'instance du **QGraphicsView** de notre visionneuse utilisée pour centrer la loupe et pour mapper des coordonnées entre scène et vue ;
- ***scene*** : l'instance de la **QGraphicsScene** ;
- ***img*** : l'instance de la **QImage** originale ;
- ***zoom*** : le facteur de zoom actuel de la pixmap affichée dans la visionneuse.

La loupe nécessite quelques attributs supplémentaires :

- ***shape*** : la forme de la loupe, "square" ou "circle" ;
- ***size*** : sa taille en pixels ;
- ***factor*** : le facteur de grossissement ;
- ***cur_pos*** : la position courante de la loupe une fois affichée ;
- ***background_color*** : la couleur de fond du **QGraphicsView** ;
- ***old_X et old_Y*** : des références de positionnement lors de déplacement.

Après les attributs d'instances il faut initialiser quelques drapeaux nécessaires au **QGraphicsItem**.

5. Les drapeaux

La liste suivante décrit les drapeaux les plus couramment utilisés.

- **QGraphicsItem.ItemIsMovable** : l'item peut être déplacé avec la souris ; si l'item comporte des items enfants, ceux-ci seront déplacés avec lui et, si l'item fait partie d'un groupe d'items sélectionnés, l'ensemble de ceux-ci sera déplacé ;
- **QGraphicsItem.ItemIsSelectable** : l'item peut être sélectionné. La sélection pourra se faire au moyen de la souris, en cliquant dessus, ou en traçant un **QRubberBand**, ou de la méthode **QGraphicsItem.setSelected()**, ou encore par **QGraphicsScene.setSelectionArea()** ;
- **QGraphicsItem.ItemIsFocusable** : l'item peut avoir le focus. Indispensable pour des interactions avec le clavier ;
- **QGraphicsItem.ItemSendsGeometryChanges** : l'item peut notifier les changements décrits par **QGraphicsItem.itemChange()**. L'utiliser peut avoir une incidence sur les performances du programme ;
- **QGraphicsItem.ItemSendsScenePositionChanges** : l'item peut notifier ses changements de position. L'usage de ce drapeau est moins réducteur en performances que **ItemSendsGeometryChanges**, car il ne notifie que les déplacements de l'item, soit **ItemScenePositionHasChanged** ;
- **QGraphicsItem.ItemIgnoresTransformations** : l'item n'hérite pas des transformations de son parent. Lorsqu'un item possède des enfants, ceux-ci subissent aussi ses transformations (rotation, dimensionnement, etc.). Avec ce drapeau l'item enfant devient insensible aux transformations de son parent. Utile si l'item contient du texte, pour garantir que celui-ci reste toujours lisible ;
- **QGraphicsItem.ItemIgnoresParentOpacity** : l'item est insensible aux changements d'opacité de son parent. Par défaut, l'opacité d'un item, définie avec **setOpacity()**, s'applique aussi à ses items enfants ;
- **QGraphicsItem.ItemDoesntPropagateOpacityToChildren** : à l'inverse du drapeau précédent, ici c'est l'item qui ne transmet pas sa valeur d'opacité à ses items enfants.

Voir les documentations respectives de PyQt (Lien 122) et PySide (Lien 123) des **GraphicsItemFlag**.
Pour la loupe, il est nécessaire que celle-ci soit déplaçable et puisse recevoir le focus.

```
magnifier.py

self.setFlags(QtGui.QGraphicsItem.ItemIsMovable |
              QtGui.QGraphicsItem.ItemIsFocusable)
```

Les dernières lignes du code de ***__init__()*** seront vues plus loin dans l'article.

Retrouvez la suite de l'article de Vincent Vande Vyvre en ligne : Lien 124

Utilisation de la LibZip

Tutoriel sur l'utilisation de la bibliothèque LibZip.

1. Introduction

Pour utiliser les archives zip plusieurs bibliothèques sont disponibles. Dans ce tutoriel nous allons nous intéresser à la libzip (Lien 125). Cette bibliothèque est écrite en C et permet de lire, créer ou modifier les archives zip. À ce jour, la dernière version est la 0.10 qui date du 18/03/2010. C'est cette version qui a été utilisée pour ce tutoriel. Le manuel de la bibliothèque est très clair et quasiment toutes les fonctions sont détaillées. Il se trouve en ligne (Lien 126) ou via man libzip pour la bibliothèque en général et man "nom de la commande".
Ce document présente comment :

- lister le contenu d'une archive zip ;
- effacer un fichier/dossier d'une archive zip ;
- renommer un fichier/dossier d'une archive zip ;
- remplacer un fichier d'une archive zip ;
- extraire un fichier/dossier d'une archive zip ;
- décompresser une archive zip ;
- créer une archive zip.

Tous les codes sont en C.
Les codes ci-dessous peuvent faire appel à des fonctions non présentes dans cette page mais le tout est disponible dans le code source C présent à la fin de la page. Le tutoriel n'est disponible que pour les systèmes UNIX/POSIX. Il ne devrait pas être difficile à adapter pour la plateforme Windows. Le code présente des appels particuliers POSIX et considère que les chemins sont de type UNIX '/' et non de type Windows '\'. Tout a été testé sur FreeBSD, si des modifications sont à apporter pour d'autres systèmes n'hésitez pas à me le faire savoir.

2. Exemple de Code

2.1. Lister le contenu d'une archive zip

2.1.1. Algorithme

Ouvrir l'archive (Fonction zip_open).
Compter le nombre d'éléments dans l'archive (Fonction zip_get_num_files).
Pour chaque élément, afficher son nom (Fonction zip_get_name).
Fermer l'archive (Fonction zip_close).

2.1.2. Code source

Afficher le contenu d'une archive Zip

```
/**
 * \fn static int afficherFichierZip(const char*
fichierZip)
 * \brief Affiche les fichiers à l'intérieur de
l'archive Zip.
 *
 * \return ZIPZAP_SUCCESS : Retour OK ;
ZIPZAP_FAILURE : Retour d'erreur.
 */
static int afficherFichierZip(const char*
fichierZip)
{

  if(fichierZip == NULL)
      return ZIPZAP_FAILURE;

  int err = 0;
  struct zip *f_zip=NULL;

  f_zip = zip_open(fichierZip, ZIP_CHECKCONS,
&err); /* on ouvre l'archive zip */

  /* s'il y a des erreurs */
  if(err != ZIP_ER_OK)
  {
      zip_error_to_str(buf_erreur, sizeof
buf_erreur, err, errno);
      printf("Error %d : %s\n",err, buf_erreur);
      return ZIPZAP_FAILURE;
  }

  /* si le fichier zip n'est pas ouvert */
  if(f_zip==NULL)
  {
      printf("Erreur à l'ouverture du fichier
%s\n", fichierZip);
      return ZIPZAP_FAILURE;
  }

  /* on récupère le nombre de fichiers dans
l'archive zip */
  int count = zip_get_num_files(f_zip);
  if(count==-1)
  {

      printf("Erreur à la lecture du fichier
%s\n", fichierZip);
      zip_close(f_zip);
      f_zip = NULL;
      return ZIPZAP_FAILURE;
  }

  int i = 0;
  printf("Nombre de fichiers dans l'archive :
%d\n",count);

  for(i=0; i<count; i++)
  {
      /* on utilise la position "i" pour
récupérer le nom des fichiers */
      printf("%s\n", zip_get_name(f_zip, i,
```

```
ZIP_FL_UNCHANGED));
  }

  zip_close(f_zip);
  f_zip = NULL;

  return ZIPZAP_SUCCESS;
}
```

2.2. Effacer un fichier/dossier d'une archive zip

2.2.1. Algorithme

Ouvrir l'archive (Fonction zip_open).
Rechercher le fichier/dossier dans l'archive (Fonction zip_name_locate).
Si c'est un dossier
    pour chaque élément du dossier, le supprimer (Fonction zip_delete).
Sinon
    le supprimer (Fonction zip_delete).
Fermer l'archive (Fonction zip_close).

2.2.2. Code source

Effacer un fichier/dossier d'une archive zip

```
/**
 * \fn static int effacerFichierZip(const char*
fichierZip, const char* fichier)
 * \brief Efface le fichier/dossier "fichier" de
l'archive zip "fichierZip".
 *
 * \param fichierZip Archive zip contenant le
fichier à effacer.
 * \param fichier Nom du fichier/dossier à
effacer.
 *
 * \return ZIPZAP_SUCCESS : Retour OK ;
ZIPZAP_FAILURE : Retour d'erreur.
 */
static int effacerFichierZip(const char*
fichierZip, const char* fichier)
{
  int visu = 0;
  struct zip * f_zip=NULL;
  int err = 0;

  f_zip=zip_open(fichierZip,ZIP_CREATE,&err);
  /* s'il y a des erreurs */
  if(err != ZIP_ER_OK)
  {
      zip_error_to_str(buf_erreur, sizeof
buf_erreur, err, errno);
      printf("Error %d : %s\n",err, buf_erreur);
      return ZIPZAP_FAILURE;
  }
  /* si le fichier zip n'est pas ouvert */
  if(f_zip==NULL)
  {
      printf("Erreur à l'ouverture du fichier
%s\n", fichierZip);
      return ZIPZAP_FAILURE;
  }

  visu=zip_name_locate(f_zip,fichier,0);
  if (visu==-1) /* recherche de l'emplacement du
fichier dans le zip */
  {
      printf("Le fichier %s n'existe pas dans
%s\n", fichier, fichierZip);
      zip_close(f_zip);
      f_zip = NULL;
      return ZIPZAP_FAILURE;

  }

  /* si on demande la suppression d'un répertoire
*/
  if(fichier[strlen(fichier)-1]=='/')
  {
      char *ptr = NULL;
      int i = 0;
      int count = zip_get_num_files(f_zip);
      if(count==-1)
      {
          printf("Erreur à l'ouverture du fichier
%s\n", fichierZip);
          zip_close(f_zip);
          f_zip = NULL;
          ptr = NULL;
          return ZIPZAP_FAILURE;
      }

      for(i=0; i<count; i++)
      {


          /* on parcourt tous les fichiers du zip
pour savoir si un fichier est dans le dossier à
supprimer */
          if( (ptr=strstr(zip_get_name(f_zip, i,
ZIP_FL_UNCHANGED), fichier)) != NULL)
          {
              /* on efface le fichier positionné
à l'index i */
              if(zip_delete(f_zip,i) == -1)
              {
                  printf("%s\n",
zip_strerror(f_zip));
                  zip_close(f_zip);
                  f_zip = NULL;
                  ptr = NULL;
                  return ZIPZAP_FAILURE;
              }
              printf("Le fichier %s a été
supprimé dans %s\n", zip_get_name(f_zip, i,
ZIP_FL_UNCHANGED), fichierZip);


          }
      }
      ptr = NULL;
  }
  else
  {
      /* sinon on supprime simplement le fichier
trouvé par zip_name_locate à la position visu */
      if(zip_delete(f_zip,visu) == -1)
      {
          printf("%s\n", zip_strerror(f_zip));
          zip_close(f_zip);
          f_zip = NULL;

          return ZIPZAP_FAILURE;
      }
      printf("Le fichier %s a été supprimé dans
%s\n", fichier, fichierZip);
  }
```

```
  /* lecture terminée, fermeture de l'archive */
  zip_close(f_zip);
  f_zip = NULL;

  return ZIPZAP_SUCCESS;
}
```

2.3. Renommer un fichier/dossier d'une archive zip

2.3.1. Algorithme

Ouvrir l'archive (Fonction zip_open).
Rechercher le fichier/dossier dans l'archive (Fonction zip_name_locate).
Si c'est un dossier
    pour chaque élément du dossier, le renommer (Fonction zip_rename).
Sinon
    le renommer (Fonction zip_rename).
Fermer l'archive (Fonction zip_close).

2.3.2. Code source

Renommer un fichier/dossier d'une archive

```
/**
 * \fn static int renommerFichierZip(const char*
fichierZip, const char* fichierFrom, const char*
fichierTo)
 * \brief Efface le fichier/dossier "fichierFrom"
en "FichierTo" de l'archive zip "fichierZip".
 *
 * \param fichierZip Archive zip contenant le
fichier à effacer.
 * \param fichierFrom Nom du fichier/dossier en
entrée.
 * \param fichierTo Nom du fichier/dossier en
sortie.
 *
 * \return ZIPZAP_SUCCESS : Retour OK ;
ZIPZAP_FAILURE : Retour d'erreur.
 */
static int renommerFichierZip(const char*
fichierZip, const char* fichierFrom, const char*
fichierTo)
{
  int visu = 0;
  struct zip * f_zip=NULL;
  int err = 0;

  f_zip=zip_open(fichierZip,ZIP_CREATE,&err);
  /* s'il y a des erreurs */
  if(err != ZIP_ER_OK)
  {
      zip_error_to_str(buf_erreur, sizeof
buf_erreur, err, errno);
      printf("Error %d : %s\n",err, buf_erreur);
      return ZIPZAP_FAILURE;
  }
  /* si le fichier zip n'est pas ouvert */
  if(f_zip==NULL)
  {
      printf("Erreur à l'ouverture du fichier
%s\n", fichierZip);
      return ZIPZAP_FAILURE;
  }

  /* recherche de l'emplacement du fichier dans
le zip */
  visu=zip_name_locate(f_zip,fichierFrom,0);
  if (visu==-1)
  {
      printf("Le fichier %s n'existe pas dans
%s\n", fichierFrom, fichierZip);
      zip_close(f_zip);
      f_zip = NULL;
      return ZIPZAP_FAILURE;

  }


  /* si on demande de renommer un répertoire */
  if(fichierFrom[strlen(fichierFrom)-1]=='/')
  {
      char *ptr = NULL;
      int i = 0;
      int count = zip_get_num_files(f_zip);
      if(count==-1)
      {
          printf("Erreur à l'ouverture du fichier
%s\n", fichierZip);
          return ZIPZAP_FAILURE;
      }
      for(i=0; i<count; i++)
      {

          /* on parcourt tous les fichiers du zip
pour savoir si un fichier est dans le dossier à
supprimer */
          if( (ptr=strstr(zip_get_name(f_zip, i,
ZIP_FL_UNCHANGED), fichierFrom)) != NULL)
          {
              /* si c'est le répertoire on lui
applique un traitement particulier */
              if(strlen(fichierFrom)==strlen(zip_
get_name(f_zip, i, ZIP_FL_UNCHANGED)))
              {
                  if(zip_rename(f_zip,i,fichierTo
) == -1)
                  {

                      printf("%s\n",
zip_strerror(f_zip));
                      zip_close(f_zip);
                      f_zip = NULL;
                      ptr = NULL;
                      return ZIPZAP_FAILURE;
                  }
                  printf("Le fichier %s a été
renommé en %s dans %s\n", fichierFrom, fichierTo,
fichierZip);
              }
              else
              {
                  /* si c'est un fichier de ce
répertoire */
                  char *buf = NULL;
                  buf =
malloc(FILENAME_MAX*sizeof(char));
                  if(buf==NULL)
                  {

                      printf("Erreur d'allocation
mémoire.\n");
                      zip_close(f_zip);
                      f_zip = NULL;
                      FREE(buf);
                      ptr = NULL;
                      return ZIPZAP_FAILURE;
                  }
```

```
                    strcpy(buf,fichierTo);

                    strcat(buf,zip_get_name(f_zip,
i, ZIP_FL_UNCHANGED)+strlen(fichierFrom));
                    printf("%s\n",buf);
                    if(zip_rename(f_zip,i,buf) ==
-1)
                    {

                        printf("%s\n",
zip_strerror(f_zip));
                        zip_close(f_zip);
                        f_zip = NULL;
                        FREE(buf);
                        ptr = NULL;


                        return ZIPZAP_FAILURE;
                    }
                    printf("Le fichier %s a été
renommé en %s dans %s\n", fichierFrom, fichierTo,
fichierZip);
                    FREE(buf);
                }
            }
        }
        ptr = NULL;
    }
    else
    {
        /* sinon on renomme simplement le fichier
trouvé par zip_name_locate à la position visu */
        if(zip_rename(f_zip,visu,fichierTo) == -1)
        {
            printf("%s\n", zip_strerror(f_zip));
            zip_close(f_zip);
            f_zip = NULL;
            return ZIPZAP_FAILURE;
        }
        printf("Le fichier %s a été renommé en %s
dans %s\n", fichierFrom, fichierTo, fichierZip);
    }



    /* lecture terminée, fermeture de l'archive */

    zip_close(f_zip);
    f_zip = NULL;

    return ZIPZAP_SUCCESS;
}
```

2.4. Remplacer un fichier d'une archive zip

2.4.1. Algorithme

Ouvrir l'archive (Fonction zip_open).
Rechercher le fichier dans l'archive (Fonction zip_name_locate).
Si le fichier n'existe pas et si on a demandé de l'ajouter dans ce cas-là
    ajouter le fichier (Fonction zip_add).
Sinon
    le remplacer (Fonction zip_replace).
Fermer l'archive (Fonction zip_close).

2.4.2. Code source

Remplacer le fichier d'une archive

```
/**
 * \fn static int remplacerFichierZip(const char*
fichierZip, const char* fichier, int ajout)
 * \brief Remplace le fichier/dossier "fichier"
dans l'archive zip "fichierZip".
 *
 * \param fichierZip Archive zip contenant le
fichier à effacer.
 * \param fichier Nom du fichier/dossier à
remplacer.
 * \param ajout Si le fichier n'existe pas dans
l'archive, on l'ajoute ou non suivant la valeur
de ajout (REPLACE_ADD ou REPLACE_NOT_ADD).
 *
 * \return ZIPZAP_SUCCESS : Retour OK ;
ZIPZAP_FAILURE : Retour d'erreur.
 */
static int remplacerFichierZip(const char*
fichierZip, const char* fichier, int ajout)
{

  /* Source de Troumad original :
http://www.developpez.net/forums/d1012322/c-
cpp/c/contribuez/faq-modifier-fichier-zip/ */
  /* modifié pour ajouter la gestion des erreurs
et quelques fonctionnalités */

  int visu = 0;
  struct zip * f_zip=NULL;
  struct zip_source * n_zip=NULL;
  int err = 0;

  f_zip=zip_open(fichierZip,ZIP_CREATE,&err);
  /* s'il y a des erreurs */
  if(err != ZIP_ER_OK)
  {
      zip_error_to_str(buf_erreur, sizeof
buf_erreur, err, errno);
      printf("Error %d : %s\n",err, buf_erreur);
      return ZIPZAP_FAILURE;
  }
  /* si le fichier zip n'est pas ouvert */
  if(f_zip==NULL)
  {
      printf("Erreur à l'ouverture du fichier
%s\n", fichierZip);
      return ZIPZAP_FAILURE;
  }
  /* on met dans le zip_source le fichier que
l'on veut remplacer */
  if((n_zip=zip_source_file(f_zip,fichier,
(off_t)0, (off_t)0)) == NULL)
  {
      printf("%s\n", zip_strerror(f_zip));
      zip_close(f_zip);
      f_zip = NULL;
      return ZIPZAP_FAILURE;
  }

  /* recherche de l'emplacement du fichier dans
le zip */
  visu=zip_name_locate(f_zip,fichier,0);
  if (visu==-1)
  {
      printf("Le fichier %s n'existe pas dans
%s\n", fichier, fichierZip);
      if(ajout==REPLACE_ADD)
      {
```

```
          /* nouveau document dans le fichier zip
: le fichier n'y est pas */
          /* c'est là qu'on fixe le nom qu'aura
le nouveau document dans le fichier zip */
          if(zip_add(f_zip,fichier,n_zip) == -1)
          {
              printf("%s\n",
zip_strerror(f_zip));
              zip_close(f_zip);
              f_zip = NULL;
              zip_source_free(n_zip);
              n_zip = NULL;
              return ZIPZAP_FAILURE;
          }
          printf("Le fichier %s a été ajouté dans
%s\n", fichier, fichierZip);
      }
      else if(ajout==REPLACE_NOT_ADD)
      {

          printf("Le fichier %s n'a pas été
ajouté ou remplacé dans %s\n", fichier,
fichierZip);



      }
      else
      {
          printf("Erreur d'utilisation de la
fonction %s\n", __FUNCTION__);



      }
  }
  else
  {
      /* modification d'un document dans le
fichier zip : le fichier est déjà dedans */
      /* notre document remplace le document qui
se trouve à l'emplacement visu */
      if(zip_replace(f_zip,visu,n_zip) == -1)
      {
          printf("%s\n", zip_strerror(f_zip));
          zip_close(f_zip);
          f_zip = NULL;
          zip_source_free(n_zip);
          n_zip = NULL;
          return ZIPZAP_FAILURE;
      }
      printf("Le fichier %s a été remplacé dans
%s\n", fichier, fichierZip);

  }

  zip_close(f_zip);
  f_zip = NULL;
  zip_source_free(n_zip);
  n_zip = NULL;

  return ZIPZAP_SUCCESS;
}
```

2.5. Extraire un fichier/dossier d'une archive zip

2.5.1. Algorithme

Ouvrir l'archive (Fonction zip_open).
Rechercher le fichier/dossier dans l'archive (Fonction zip_name_locate).
Si c'est un dossier
pour chaque élément du dossier, l'extraire (Fonction extraireFichier -> Fonctions zip_stat, zip_fopen, zip_fread, zip_fclose).
Sinon
l'extraire (Fonction extraireFichier -> Fonctions zip_stat, zip_fopen, zip_fread, zip_fclose).
Fermer l'archive (Fonction zip_close).

2.5.2. Code source

Voir code source en ligne : Lien 127

2.6. Décompresser une archive zip

2.6.1. Algorithme

Ouvrir l'archive (Fonction zip_open).
Compter le nombre d'éléments dans l'archive (Fonction zip_get_num_files).
Extraire chaque élément (Fonction extraireFichier -> Fonctions zip_stat, zip_fopen, zip_fread, zip_fclose).
Fermer l'archive (Fonction zip_close).

2.6.2. Code source

Décompresser une archive

```
static int decompresserFichierZip(const
char* fichierZip, const char*
repSortie, const char* motDePasse)
{

  /* Code d'origine : */
  /* http://www.siteduzero.com/forum-83-372372-
p1-c-zip-et-unzip.html#r3519929 */
  /* http://forum.ubuntu-fr.org/viewtopic.php?
pid=2596893#p2596893 Visité le 02/01/2011 */
  /* modifié pour ajouter la gestion des erreurs
et quelques fonctionnalités */

  int err=0;
  struct zip *f_zip=NULL;



  /* 1. Ouverture de l'archive */
  f_zip = zip_open(fichierZip, ZIP_CHECKCONS,
&err);
  /* s'il y a des erreurs */
  if(err != ZIP_ER_OK)
  {
      zip_error_to_str(buf_erreur, sizeof
buf_erreur, err, errno);
      printf("Error %d : %s\n",err, buf_erreur);
      return ZIPZAP_FAILURE;
  }
  /* si le fichier zip n'est pas ouvert */
  if(f_zip==NULL)
  {
      printf("Erreur à l'ouverture du fichier
%s\n", fichierZip);
      return ZIPZAP_FAILURE;
  }

  /* 2. on récupère le nombre de fichiers dans
l'archive zip */
  int count = zip_get_num_files(f_zip);
  if(count==-1)
```

```
  {
      printf("Erreur à l'ouverture du fichier
%s\n", fichierZip);
      zip_close(f_zip);
      f_zip = NULL;
      return ZIPZAP_FAILURE;
  }

  printf("Nombre de fichiers dans l'archive :
%d\n", count);

  int i;

  /* 3. on lit tous les fichiers */
  for(i=0; i<count; i++)
  {
      /* on lance la fonction qui extraira le
fichier en position "i" dans l'archive Zip */
      if(extraireFichier(i, f_zip, repSortie,
motDePasse)==ZIPZAP_FAILURE)
      {
          printf("Erreur à l'extraction du
fichier %s\n", zip_get_name(f_zip, i,
ZIP_FL_UNCHANGED));
          zip_close(f_zip);
          f_zip = NULL;
          return ZIPZAP_FAILURE;
      }
  }

  /* lecture terminée, fermeture de l'archive */
  zip_close(f_zip);
  f_zip = NULL;


  return ZIPZAP_SUCCESS;
}
```

2.7. Créer une archive zip

2.7.1. Algorithme

Créer l'archive suivant le mode choisi (Fonction zip_open).
Vérifier que le fichier/dossier n'est pas déjà présent dans l'archive (Fonction verfiPresenceFichier).
Si on ajoute un dossier
on l'ajoute à l'archive (Fonction zip_add_dir),
on ajoute tous ses éléments dans l'archive (Fonction ajouteDossier ->Fonctions zip_source_file, zip_add_dir, zip_add).
Sinon
on ajoute le fichier à l'archive (Fonctions zip_source_file, zip_add).
Fermer l'archive (Fonction zip_close).

2.7.2. Code source

Créer une archive

```
/* Ajoute les fichiers tels qu'ils sont
envoyés dans la ligne de commande */
/* Pose problème avec les fichiers du style
"../../fichiers" */
/**
 * \fn static int creerFichierZip(const char*
fichierZipEnSortie, int modeZip, int argc, char
*arg[])
 * \brief Créer l'archive zip
"fichierZipEnSortie" suivant le mode "modeZip"
avec les fichiers contenus dans *arg[].
 *
 * \param fichierZipEnSortie Le nom de l'archive
zip à créer en sortie.
 * \param modeZip Mode de création de l'archive
ADD ; CREATE ou EXIST. (voir les définitions).
 * \param argc Nombre de fichiers reçus par le
programme et transmis à la fonction.
 * \param argv Chaînes de caractères reçues par
le programme et transmises à la fonction. Ces
chaînes contiennent les fichiers/répertoires à
ajouter.
 *
 * \return ZIPZAP_SUCCESS : Retour OK ;
ZIPZAP_FAILURE : Retour d'erreur.
 */
static int creerFichierZip(const char*
fichierZipEnSortie, int modeZip, int argc, char
*arg[])
{

  struct zip *f_zip=NULL;
  struct zip_source *doc = NULL;
  int err = 0;

  /* Suivant le mode choisi pour la création de
l'archive on définit les flags */
  if(modeZip==CREATE)
  {
      /* on supprime l'archive pour en créer une
nouvelle */
      remove(fichierZipEnSortie);
      f_zip=zip_open(fichierZipEnSortie,
ZIP_CREATE, &err); /* on crée l'archive */
  }
  else if(modeZip==EXIST)
  {
      f_zip=zip_open(fichierZipEnSortie,
ZIP_EXCL|ZIP_CREATE, &err); /* on ne crée pas
l'archive si elle existe, Jacques THERY2011-12-
06T12:28:12.39Vous ne pensez pas que la phrase
qui suit n'a pas d'utilité ?mais si elle n'existe
pas on la crée */
  }
  else if(modeZip==ADD)
  {
      f_zip=zip_open(fichierZipEnSortie,
ZIP_CREATE, &err); /* on crée l'archive si elle
n'existe pas */
  }
  else
  {
      printf("Erreur du choix du mode\n");
      usage(arg[0]);
      return ZIPZAP_FAILURE;
  }

  /* s'il y a des erreurs */
  if(err != ZIP_ER_OK)
  {
      zip_error_to_str(buf_erreur, sizeof
buf_erreur, err, errno);
      printf("Error %d : %s\n",err, buf_erreur);
      return ZIPZAP_FAILURE;
  }
  /* si le fichier zip n'est pas ouvert */
  if(f_zip==NULL)
  {
      printf("Erreur à l'ouverture du fichier
%s\n", fichierZipEnSortie);
      return ZIPZAP_FAILURE;
```

```
  }

  /* on positionne i à 3 car on a dans le tableau
: "0. Le nom du programme" "1. Le mode de
création" "2. Le nom de l'archive" */
  int i = 3;
  for(; i<argc; i++)
  {

      if(verifPresenceFichier(f_zip,
arg[i])==ZIPZAP_FAILURE)
      {
          printf("Erreur lors de l'ajout du
fichier/dossier \"%s\" à l'archive %s\n", arg[i],
fichierZipEnSortie);
          zip_close(f_zip);
          f_zip = NULL;
          zip_source_free(doc);
          doc = NULL;
          return ZIPZAP_FAILURE;
      }

#if defined(__POSIX_VISIBLE) ||
defined(_POSIX_C_SOURCE)
      /* on teste si le fichier demandé est un
dossier ou non */
      struct stat f_stat;
      if(lstat(arg[i], &f_stat) != -1)
      {
          if(S_ISDIR(f_stat.st_mode))
          {
              /* Si cela en est un, on l'ajoute à
l'archive */
              if(zip_add_dir(f_zip, arg[i]) ==
-1)
              {

                  printf("%s\n",
zip_strerror(f_zip));
                  zip_close(f_zip);
                  f_zip = NULL;
                  zip_source_free(doc);
                  doc = NULL;

                  return ZIPZAP_FAILURE;
              }

              /* Puis on ajoute tous les fichiers
du dossier et de ses sous-dossiers */
              if(ajouteDossier(arg[i],
f_zip)==ZIPZAP_FAILURE)
              {
                  printf("Erreur à l'ajout du
dossier %s\n", arg[i]);
                  zip_close(f_zip);
                  f_zip = NULL;
                  zip_source_free(doc);
                  doc = NULL;
                  return ZIPZAP_FAILURE;
              }


          }
          else /* Sinon on le traite comme un
fichier normal */
          {

              /* que l'on récupère dans un
zip_source */
              if((doc=zip_source_file(f_zip,arg[i
],(off_t)0,(off_t)0))==NULL)
              {

                  printf("%s\n",
zip_strerror(f_zip));
                  zip_close(f_zip);
                  f_zip = NULL;
                  zip_source_free(doc);
                  doc = NULL;
                  return ZIPZAP_FAILURE;
              }

              /* pour l'inclure dans l'archive */
              if(zip_add(f_zip, arg[i], doc)==-1)
              {

                  printf("%s\n",
zip_strerror(f_zip));
                  zip_close(f_zip);
                  f_zip = NULL;
                  zip_source_free(doc);
                  doc = NULL;
                  return ZIPZAP_FAILURE;
              }


          }
      }
      else
      {
          printf("Le fichier %s n'existe pas\n",
arg[i]);
          zip_close(f_zip);
          f_zip = NULL;
          zip_source_free(doc);
          doc = NULL;
          return ZIPZAP_FAILURE;
      }

#endif /* posix */
      /* TODO : la même chose pour Windows */

  }

  zip_close(f_zip);
  f_zip = NULL;
  zip_source_free(doc);
  doc = NULL;

  return ZIPZAP_SUCCESS;
}
```

3. Conclusion

Les codes sources présentés ci-dessus montrent comment il est facile avec la LibZip de réaliser les opérations classiques sur des archives. Je vous recommande l'utilisation de cette bibliothèque très puissante et très simple d'utilisation. Elle est d'ailleurs utilisée par de nombreux gros projets et ce n'est sans doute pas pour rien.
Vous pouvez télécharger le code source complet et la Makefile dans cette archive zip créée avec les exemples du code.
Télécharger ZipZap : Lien 128.

Retrouvez l'article de Loïc Bartoletti en ligne : Lien 129.

Le débogage sous Visual Basic 6 & Visual Basic pour Application (1re partie)

Tout ce que vous devez savoir sur le débogage et la gestion des erreurs sous Visual Basic 6.

1. Introduction

La lecture du forum Visual Basic laisse à penser que beaucoup d'utilisateurs ne connaissent pas ou mal certaines possibilités de l'IDE VB6 concernant le traitement des erreurs ni l'éventail des possibilités offertes pour le débogage : il m'a donc semblé utile de rédiger ce tutoriel sommaire concernant les spécificités de Visual Basic 6 dans ce domaine.

Pour tout complément d'information, consultez la MSDN : Lien 130.

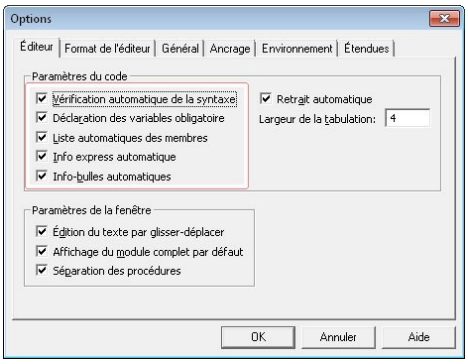
2. Les outils

L'IDE Visual Basic est riche d'outils d'assistance qui permettent notamment l'aide à l'écriture, son contrôle, la gestion personnalisée du mode runtime ou la possibilité d'interactions, etc.

2.1. Les outils d'aide à l'écriture

Menu Outils/Options (Alt OT) > Onglet Éditeur

L'IDE Visual Basic dispose avec la technologie Intellisense de différents outils d'aide à l'écriture (Listes automatiques , Info-bulles, des Infos-express). Outre le fait de limiter le risque d'erreurs de syntaxe, ils ont pour propriété de devenir non fonctionnels s'il existe une erreur (défaut de typage, de paramètres, etc.) **ce qui permet donc indirectement de signaler une erreur.**



Cocher toutes les options «Paramètres du code».

2.1.1. Vérification automatique de la syntaxe

La vérification automatique de la syntaxe permet de vérifier la syntaxe à chaque ligne de code.

2.1.2. Déclaration des variables obligatoire

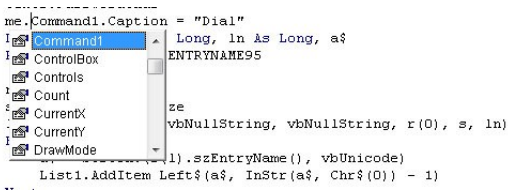
Cocher systématiquement l' « Option Explicit » force à déclarer chaque variable en ajoutant une instruction «Option Explicit» en tête de module au moment de sa création.
L'absence de typage étant une source importante d'erreurs, il est recommandé d'effectuer le typage de chaque variable (il est même recommandé de pratiquer un typage fort => non-utilisation du type Variant).

```
Option Explicit

'- Exemples de ce qu'il faut faire et NE PAS faire :

Dim MyVar     ' INCORRECT: MyVar est de type Variant
Dim MyVar As String     ' CORRECT: Typage fort, MyVar est définie comme type String

Dim MyVar1, MyVar2 As Integer     ' INCORRECT: seule MyVar2 est typée comme Integer, MyVar1 est de type Variant
Dim MyVar1 As Integer, MyVar2 As Integer     ' CORRECT: MyVar1 ET MyVar2 sont correctement typées comme Integer
```

Vous devrez ajouter manuellement *Option Explicit* dans tous les modules de votre projet, créés ou importés **AVANT** d'avoir coché cette case ! (Form, Module, Module de classe, etc.)

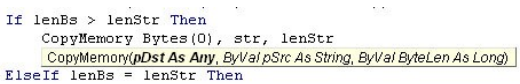
2.1.3. Autocomplétion

L'autocomplétion (liste automatique des membres) fournit au développeur dans une zone de liste les propriétés, méthodes, etc. du modèle objet sélectionné (racine) ce qui permet de réduire le temps de développement et assure un code syntaxiquement correct (caractère d'activation « . »).



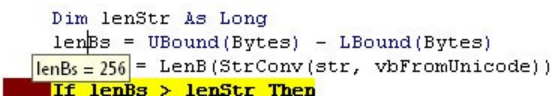
2.1.4. Info-express automatique

La zone d'Info-express (Ctrl+Maj+I/Ctrl+I) est complémentaire de l'autocomplétion ; elle fournit la syntaxe de fonction, d'instruction, de méthode et des paramètres liés en indiquant le paramètre en cours d'édition.



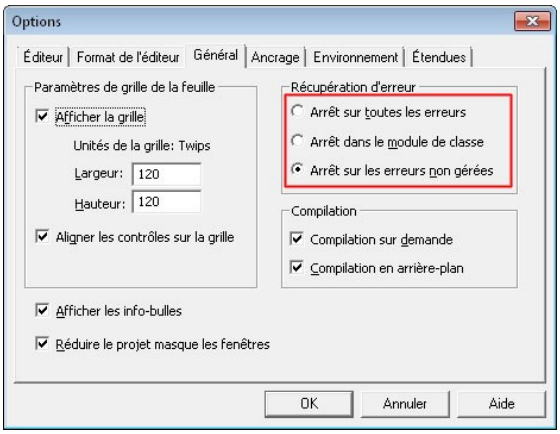
2.1.5. Info-bulle automatique

L'outil Info-bulle fonctionne en mode arrêt ; il permet d'afficher (dans la limite des 72 premiers caractères - nom de variable inclus) la valeur en cours de l'expression se situant sous le curseur.



2.2. Le paramétrage des Options de gestion d'erreurs

Le paramétrage des options de l'IDE va conditionner le comportement des gestionnaires d'erreurs notamment l'activation ou la désactivation des gestionnaires.



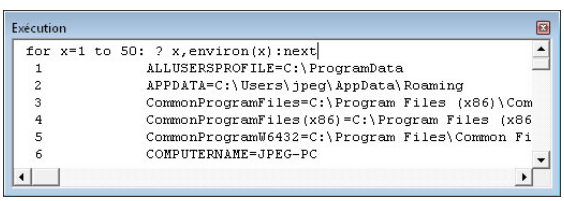
Trois possibilités sont présentes sur le comportement face à une erreur :

- arrêt sur toutes les erreurs : désactive tous les gestionnaires d'erreurs et marque un arrêt sur toutes les erreurs ;
- arrêt sur les erreurs non gérées : active les gestionnaires d'erreurs et marque un arrêt uniquement sur les erreurs non gérées ;
- arrêt dans les modules de classe : se distingue de l'option précédente marquant un point d'arrêt dans le module de classe au lieu du module d'appel de la classe.

2.3. Les outils dédiés au débogage

La fenêtre d'exécution (Ctrl+G) permet les actions suivantes :

- exécuter directement des instructions (pour une utilisation d'instructions multiples, utilisez une seule ligne et le séparateur d'instructions « : ») :



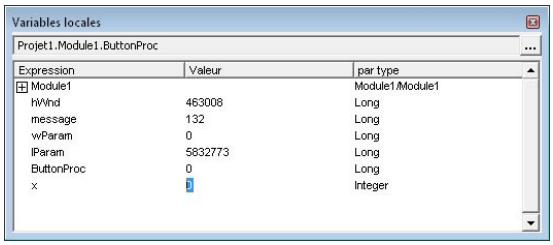
- visualiser le résultat de partie de code en cours d'exécution en mode runtime (instruction Debug.Print) :

```
resp = RasDial(ByVal 0, ByVal 0, rp, 0, ByVal 0, h)
Debug.Print "resp", resp
```

- lancer une procédure particulière quand le code actif s'exécute et se trouve en mode arrêt.

L'utilisation de la fenêtre d'exécution est utilisable en mode arrêt lors d'une exécution en mode runtime ou directement en mode conception.

La fenêtre des variables locales (VB:Alt+I+L - VBA: Alt+A+V)

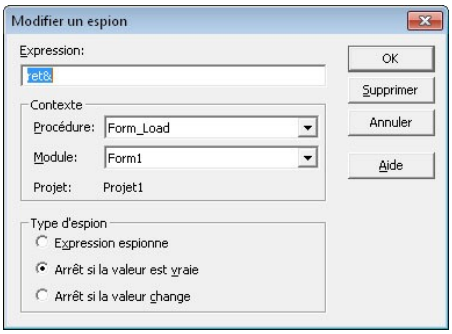


Elle permet de **visualiser** ou **modifier** les variables en cours.
La capacité d'affichage d'une variable est fonction de la largeur maximum d'affichage de la fenêtre.

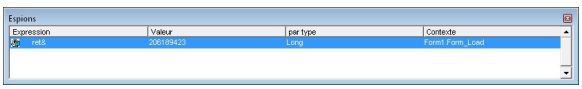
Pour modifier la valeur d'une variable dans la fenêtre Variables locales : double-cliquer sur la valeur concernée afin de l'éditer.

La fenêtre Espions

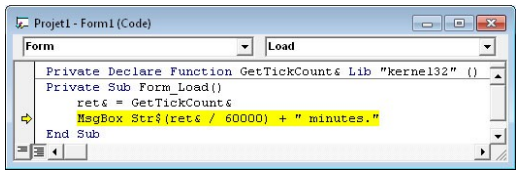
Permet de suivre la valeur de retour de propriétés, de fonctions, etc. ou de déterminer une expression dont la valeur stoppera l'exécution du code en mode runtime.
L'affichage des variables est limité aux 250 premiers caractères.
Les menus en rapport : Ajouter un espion - Modifier un espion



génèrera la fenêtre suivante en cours d'exécution...



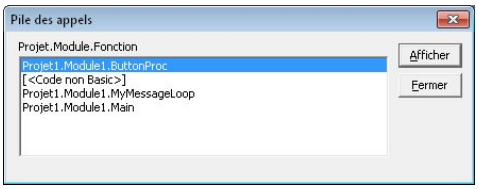
et un arrêt sur la ligne suivante.



Il est possible d'ajouter un espion en effectuant un Glisser Déposer de la variable préalablement sélectionnée dans la fenêtre Espions.

Pile des Appels (Ctrl+L)

Cette fenêtre affiche la liste des procédures en cours d'exécution pendant le mode arrêt et éventuellement affiche le code de la procédure sélectionnée.
Cet outil est utile pour analyser les procédures récursives ou pour retrouver un gestionnaire d'erreur parent.



2.4. Les utilitaires dédiés

Les outils fournis par MZ-Tools : Lien 131.

L'addin MZ-Tools met à disposition deux outils intéressants pour la gestion d'erreurs :

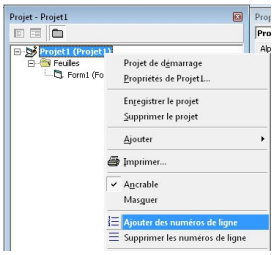
Le numéroteur de lignes

Cet outil permet d'automatiser l'indexation des lignes de code

- soit au niveau procédure ;
- soit au niveau d'un module ;
- soit au niveau de tout le projet.

Il est possible de déterminer dans le paramétrage de l'addin, l'incrémentation des numéros de lignes.

La numérotation au niveau projet ou module est possible en utilisant le menu contextuel (clic droit) après sélection du contexte dans l'Explorateur de projet.



Ajouter une procédure d'erreur

Il peut être fastidieux d'ajouter une gestion d'erreur à de nombreuses procédures ; cet outil répond à ce problème en suivant un modèle prédéfini en fonction de différentes variables.

Les champs disponibles pour définir le modèle sont :

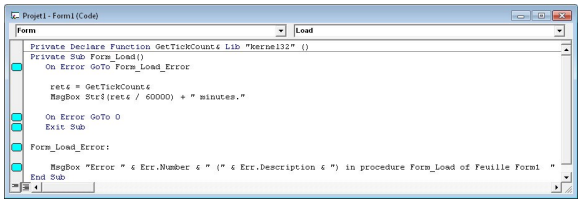
Variables	Description	Équivalence
{PROJECT_ FILENAME}	Nom du fichier de projet	VBProject.FileName
{PROJECT_ NAME}	Nom du projet	VBProject.Name
{MODULE_ TYPE}	Type de module	VBComponent.Type
{MODULE_ FILENAME}	Nom de fichier du module	VBComponent.FileName
{MODULE_ NAME}	Nom du module	VBComponent.Name
{FUNCTION_ RETURN_TYPE_ NAME}	Nom du type de retour de la procédure	ex : Long, Boolean, etc.
{FUNCTION_ RETURN_TYPE_ PREFIX}	Préfixe (1re lettre) du type de retour de la procédure	
{PROCEDURE_ TYPE}	Type de la procédure	Member.Type
{PROCEDURE_ NAME}	Nom de la procédure	Member.Name
{PROCEDURE_ BODY}	Corps de procédure (représente le code de la procédure après le premier bloc de déclaration)	Lines

Le modèle est à définir dans l'interface des Options MZ-Tools (onglet «Gestionnaire d'erreur»).

Soit, par exemple, le modèle suivant :

```
On Error GoTo {PROCEDURE_NAME}_Error

{PROCEDURE_BODY}

On Error GoTo 0
Exit {PROCEDURE_TYPE}

{PROCEDURE_NAME}_Error:

MsgBox "Error " &amp; Err.Number &amp; " (" &amp;
Err.Description &amp; ") in procedure
{PROCEDURE_NAME} of {MODULE_TYPE} {MODULE_NAME}"
```

qui génèrera le résultat suivant :



N. B. Les lignes ajoutées sont marquées d'un signet dans la marge.

L'outil «Signet» permet de signaler une ligne en ajoutant un tag dans la marge.

3. Gestion des erreurs par le code

3.1. Les instructions de gestion

On Error ... *Instruction*

L'instruction On Error supporte plusieurs syntaxes :

- On Error Goto *Etiquette* : permet de rerouter l'exécution vers un bloc de gestion de l'exception levée, placé en général immédiatement après une instruction Exit Sub/Exit Function/Exit Property ;
- On Error Goto 0 : supprime toute gestion d'erreur préalablement installée ;
- On Error Resume Next : permet de générer un gestionnaire d'erreurs «passif» : le contrôle de l'exécution est transmis directement à la ligne suivante sans qu'aucun traitement ne soit effectué.

Utiliser impérativement une variable intermédiaire si une instruction de test doit être effectuée sous le contrôle d'une instruction On Error Resume Next.

```
' exemple :
On Error Resume Next
ret= Val(myString)
If ret<>0 Then
    '.../...
endif

' et non pas :
If Val(myString) Then
```

La portée d'une instruction On Error ... est limitée à la procédure courante.

Resume *Instruction*

Provoque une réinitialisation de l'objet Err (équivalent à Err.Clear) et la poursuite de l'exécution après la gestion de l'exception :

- Resume : l'exécution reprend à la ligne où l'interruption a eu lieu ;
- Resume Next : l'exécution reprend à la ligne suivante en ignorant la ligne ayant levé l'exception ;
- Resume *Line* : l'exécution reprend à une ligne spécifique en fonction d'un numéro de ligne ou d'une étiquette.

Error *Instruction*

Assure la compatibilité avec les versions précédentes ; préférer *Err.Raise*.

IsError *Expression*

Renvoie *Vrai* si *expression* est une valeur d'erreur.

CVErr *Function*

Permet de créer des erreurs définies par l'utilisateur dans des procédures créées par l'utilisateur (voir également la constante vbObjectError).

Err *Object*

La portée de l'objet Err est limitée à la procédure à laquelle est associé un gestionnaire d'erreurs «On Error...».

Propriétés	Type	Lect/Ecr.	Description
Description	String	RW	Description d'un objet Err
Number	Long	RW	ID
Source	String	RW	Nom de l'objet (module) ou de l'application à l'origine de l'erreur.
HelpContext	String	RW	Id du contexte d'aide associé.
HelpFile	String	RW	Chemin vers un fichier d'aide
LastDllError	Long	R	Dernier code d'erreur produit par un appel à DLL

Méthodes	Description
Clear	Réinitialise l'objet Err
Raise	Génère une erreur

ERL *Function*

Retourne le n° de ligne (Long) sur laquelle s'est produite la dernière erreur levée (portée procédure).

Debug *Object*

Méthodes	Description
Print	Retourne la valeur d'une expression dans la fenêtre d'exécution quand le code s'exécute en mode runtime
Assert	Arrête l'exécution en mode runtime sur la ligne concernée quand la valeur de l'expression est fausse (ignorée en mode compilé)

3.2. Créer un gestionnaire d'erreurs

La création d'un gestionnaire d'erreurs «actif» consiste à élaborer une gestion au cas par cas afin de répondre à toutes les erreurs susceptibles de se produire.
Illustration par l'exemple...

```
Option Explicit

' Nécessite que l'option de l'Ide «Arrêt sur les
erreurs non gérées» soit activée
Private Sub Form_Load()
    Dim strErrnum As String
```

```
    Dim intErr    As Integer
    Dim intReturn As Integer

    On Error GoTo Exemple1_Error

10    intErr = 3 / intReturn     ' (div 0)
20    intErr = 10 ^ 31           ' dépassement de
capacité

30    strErrnum = 71             ' disque non prêt
40    intErr = Val(strErrnum)
50    Err.Raise Number:=intErr

Exit Sub

Exemple1_Error:
    Select Case Err.Number
        Case 6
            Resume Next ' Erreur non corrigée :
Passer à la ligne suivante

        Case 11
            intReturn = 1 ' Correction de
l'erreur ...
            MsgBox "=> La valeur 0 va être
remplacée par la valeur 1"
            Resume ' Reprendre l'exécution

        Case 68
            If MsgBox(Err.Description,
vbRetryCancel) = vbRetry Then Resume ' Exécuter à
nouveau ou annuler ?

        Case Else       ' Signaler toute autre
erreur et sortie du code de la procédure
incriminée
            MsgBox Err.Description ,,  "Form_Load
(Line " & Erl & ")"

    End Select
End Sub
```

3.3. Le choix de l'externalisation

Le choix du mode de sortie de l'erreur dépend du contexte :

- la boîte de dialogue (Msgbox) : pour interagir avec l'utilisateur final ou permettre d'entrer en mode pas à pas pendant la mise au point ;
- la fenêtre d'exécution (Debug.Print) : permet de signaler une erreur sans interrompre le déroulement du programme ;
- le fichier journal : il a le mérite de ne pas être éphémère, de transmettre des informations non déformées et de pouvoir être traité automatiquement - à réserver en production ou dans le cadre de mise au point suite à erreur fatale (dans ce cadre, l'écriture doit être optimisée) ;
- le presse-papier : ce peut être une solution intermédiaire entre la fenêtre de Debug et le fichier log pour la phase de mise au point ;
- la console : peut éventuellement présenter un intérêt lorsqu'une quantité importante d'information doit être présentée à l'utilisateur final ;
- l'imprimante : accessoirement, peut être utilisée pour analyser la génération de résultats erronés.

3.4. Gestionnaire d'erreurs centralisé

Établir un gestionnaire centralisé consiste à réaliser une ou plusieurs procédures spécifiques de traitement des exceptions qu'il suffira ensuite d'appeler lors de l'interception de l'erreur.

Comme pour le choix du mode de sortie, c'est le contexte qui déterminera le choix entre gestionnaire en ligne et/ou gestionnaire centralisé :

- un gestionnaire centralisé présente l'avantage de ne pas multiplier les actions similaires, en conséquence, sa modification s'en trouve simplifiée puisque répercutée sur l'ensemble de l'application ; c'est même un investissement si le gestionnaire est réutilisable dans des projets différents ;
- un gestionnaire en ligne est, au contraire, destiné aux cas particuliers.

L'un comme l'autre peuvent éventuellement dépendre d'un fichier de ressources dédié qui sera aisément traductible (ex. vb6xx.dll pour Visual Basic).

4. Spécificités en mode compilé

Uniquement Visual Basic.

4.1. Les options et paramètres de compilation

> Propriétés du projet (Alt PP) > Onglet Compilation > Options avancées

Conditionnent le comportement de l'exécutable compilé face à une exception.
Si elles permettent d'optimiser la vitesse d'exécution en supprimant des contrôles internes, leur utilisation risque également de créer de graves dysfonctionnements pour peu que la gestion d'erreurs ne soit pas des plus strictes.

Néophytes, s'abstenir !!

4.2. Débogage symbolique

> Propriétés du projet (Alt PP) > Onglet Compilation

Générer les informations de débogage symbolique :
cette option permet de générer automatiquement à la compilation un fichier pdb utilisable notamment par VC++ afin de déboguer l'exécutable préalablement compilé en mode natif.
Ces informations sont générées chaque fois qu'un fichier OBJ est créé par le compilateur ; elles contiennent les informations de type, de prototype des fonctions destinées au débogueur.

4.3. Remarques

Le déploiement et l'installation d'un exécutable nécessitent de respecter quelques règles incontournables pour éviter certaines erreurs liées à l'installation.

- Lorsque qu'un exécutable inclut des dépendances ActiveX, il importe de conserver les fichiers dep associés ainsi que les dépendances avec les sources, car ils décrivent les éventuelles

dépendances imbriquées qu'il sera ultérieurement difficile d'identifier sans eux.
- Il est essentiel de réaliser (et de conserver avec les sources) un Setup d'installation car il n'est pas acquis que la version du système d'exploitation sur lequel l'installation s'effectuera distribue les dépendances du système sur lequel a été compilé l'exécutable (par exemple, Vista/Seven ne distribue pas une partie des DLL/OCX distribuées sous Win2000/XP).
- Rappelez à l'utilisateur qu'il est essentiel avant l'installation de fermer les autres programmes et d'effectuer l'installation en mode administrateur.
- Il est nécessaire de tester l'exécutable sous les différents OS disponibles afin de connaître les éventuelles contraintes d'installation ou incompatibilités (par exemple, installation en mode compatible sous Seven 32 ou 64 bits).

L'Observateur d'évènements de Windows et les journaux liés peuvent être une source d'information complémentaire sur les erreurs.

5. Le débogage

5.1. Mise au point d'un programme

5.1.1. L'exécution en mode runtime

Tout contrôle d'un exécutable commence par une exécution en mode runtime...
Personne n'imaginerait compiler directement un exécutable et le distribuer sans cette étape intermédiaire.

L'IDE VB6 permet d'exécuter du code en mode «runtime» **tout en permettant d'intervenir sur celui-ci lors de son exécution** dès qu'il entre en mode arrêt.

Cette propriété essentielle permet :
- la modification du code existant ;
- l'ajout de lignes d'instructions ;
- l'ajout ou la modification de déclarations ;
- la modification de valeur de variables ;
- le saut d'instructions, de procédures, etc.

La réunion de ces possibilités laisse imaginer la puissance offerte pour la mise au point.

Il est possible dans le même temps, **d'exécuter du code en parallèle depuis la fenêtre d'Exécution**
=> l'exécution des instructions, procédures ou fonctions se répercutera sur le code global en mode arrêt.

L'usage de cette technique a toutefois des limites :
- il faut l'éviter dans les procédures de rappels au risque d'un crash de l'IDE ;
- il n'est pas question d'effectuer des modifications importantes dans ce contexte car la stabilité de l'IDE est en relation avec le cumul des erreurs interceptées, leur gravité et les modifications opérées consécutivement.

5.1.2. Tester la stabilité

Tester la stabilité (la robustesse) d'un exécutable est également indispensable car, si un code s'exécute sans problème jusqu'à son terme ou a contrario permet de mettre en évidence les erreurs, rien ne prouve qu'il s'exécutera correctement en condition extrême (charge CPU, saturation mémoire, etc.).
Outre des outils professionnels dédiés à cet usage, il est possible d'exécuter un test de charge simple qui consiste à appeler consécutivement une même routine (ou un ensemble de routines) un grand nombre de fois - on observera en parallèle l'utilisation des ressources.

Exemple :

```
Dim x   As Long, ret As Long
Const k As Long = 1000

On Error Goto cath_Err

For x=1 To k
    ret = myFunction()
Next

cath_Err:
    Debug.Print ret
    If x<=k Then Debug.Print "Error at " & k & "
cycles"
```

Un second test simple consiste à renouveler plusieurs fois la routine précédente et comparer le résultat final retourné à chaque test.

5.1.3. Erreur inattendue

C'est l'erreur qui se produit lors d'une nième exécution, voire même aléatoirement sans qu'on puisse de premier abord en identifier l'origine.

On pourra tenter d'identifier si l'environnement pose problème - il faudra dans ce cadre :
- travailler avec un environnement restreint (fermeture de tous les programmes, désactivation des éventuels compléments, etc.) ;
- si nécessaire, utiliser une session de l'OS en mode sans échec ;
- accessoirement, tester si l'erreur se reproduit sous différents systèmes d'exploitation.

5.2. Exécuter le code

Il existe deux modes d'exécution :
- le mode Pas à pas (F8) qui permet d'exécuter le programme ligne à ligne,
- le mode normal (F5) qui exécute le programme d'une traite jusqu'à ce qu'une erreur ou un point d'arrêt soit éventuellement rencontré.

Les deux modes peuvent être utilisés consécutivement au sein d'une même session.

Une fois en mode arrêt, il est possible d'exécuter le code par tranche en utilisant :
- mode « Pas à pas principal » **(Maj+F8)** ;
- mode « Pas à pas sortant » **(Ctrl+Maj+F8)** ;
- « Exécuter jusqu'au curseur » **(Ctrl+F8)**.

Arrêter l'exécution sur une ligne particulière :

- le point d'arrêt (clic dans la marge de la ligne concernée) : il permet de stopper l'exécution juste avant l'exécution d'une ligne précise,
- l'instruction Stop est identique au point d'arrêt,
- en fonction d'un test dans le code (cf. l'instruction Debug.Assert),
- arrêts conditionnels en fonction d'un Espion (cf. Outils).

L'instruction Stop ne doit pas apparaître dans un exécutable compilé.

=> Durant la phase de mise au point, il peut être intéressant d'utiliser les arguments de compilation conditionnelle
(VB: Alt PP > onglet Créer - VBA: Utiliser des constantes conditionnelles « #Const ... »)
afin de gérer automatiquement la non-compilation des instructions d'aide au débogage (Stop, Msgbox, Debug.Print, etc.).

exemple VB : « Arguments de compilation conditionnelle » > *NOCOMPILATION=1*
exemple VBA : *#Const NOCOMPILATION=1*

```
#If NOCOMPILATION Then
    Stop
#End If
```

5.3. Cas particuliers des exécutions d'instances

Uniquement Visual Basic

Si l'exécution en mode runtime d'un exécutable standard (Exe) ne pose pas de problèmes particuliers, il est moins évident d'exécuter le code des DLL ActiveX, controls et Addins
qui ne peut être exécuté directement dans l'IDE puisqu'une instance doit être créée.
Leur débogage nécessite donc quelques artifices...

Utilisez le paramétrage : (Outils/Options> Onglet Général) : « Arrêt sur les erreurs non gérées » ou « Arrêt dans les modules de classe ».

5.3.1. Control ActiveX

- Créer un nouveau projet Exe Standard (Ctrl N) ; celui-ci servira de projet support pour le test.
- Créer un groupe de projet en ajoutant le projet du contrôle à tester (Alt FA) => les contrôles sont disponibles dans la Boite à Outils du projet de test.

Après avoir placé si nécessaire, un point d'arrêt à l'endroit où commencer l'exécution en mode Pas à pas (ligne par ligne)

- déposer le contrôle à tester sur la feuille du projet de test => la partie « Concepteur » du code s'exécute...
- exécutez le projet de test en mode Runtime (F5) => la partie du code « Utilisateur » s'exécute...

L'ordre d'ouverture des projets dans le groupe détermine le projet qui s'exécutera en mode runtime.

5.3.2. DLL ActiveX

Pratiquer comme pour un Control en utilisant un groupe de projets, mais en ajoutant une référence (Alt PR) au projet dans le projet de test.
=> Une entrée *«Nom_du_projet» (Chemin d'accès : Chemin_du_projet.vbp)* a été ajoutée à la liste des références disponibles.

Il existe toutefois une autre solution à l'utilisation d'un groupe de projets :

- ajouter un module standard au projet à tester ;
- ajouter une procédure de test dans laquelle est initialisée la classe à tester ;
- exécuter la procédure en l'appelant depuis la fenêtre d'Exécution.

5.3.3. AddIn

L'exécution de l'addin est lancée en mode normal depuis le projet de l'addin (F5) : l'instance en cours se met en attente de connexion
=> la connexion à l'instance en cours s'effectuera en ouvrant une instance de l'application liée.

Afin de ne pas avoir à gérer l'installation de l'addin à l'ouverture de l'application cliente :
=> Définir le mode de démarrage de l'addin (Comportement Initial) à « **Startup** ».

6. Conclusion

La gestion des erreurs est un vaste sujet, incontournable pour toute application, et les outils de mise au point sous VB6 sont nombreux.
En faire le tour nécessiterait la rédaction d'un ouvrage- les lecteurs désireux d'approfondir le sujet peuvent toutefois trouver plus d'information en consultant la MSDN.

Note aux utilisateurs VB : il existe également un projet de démo «Errors.vbp» illustrant la gestion d'erreurs dans le répertoire de la MSDN
(path_to_MSDN\MSDN98\98VSa\1036\SAMPLES\VB98\Errors\ERRORS.VBP).

Retrouvez l'article de DarkVader en ligne : Lien 132

Pourquoi générer le code JavaScript est une fausse bonne idée.

Bonjour.

En lisant le forum et en répondant aux questions, je me suis dit qu'une petite mise au point semblait nécessaire.

Souvent, dans nos développements, nous avons un serveur dynamique PHP, ASP, JAVA, C#, Ruby, etc.

Il est très facile et tentant de générer les divers éléments dont on a besoin directement avec le langage de ce serveur.

C'est simple, ça apporte des facilités mais ce n'est pas toujours aussi efficace qu'on le pense.

Lorsqu'on crée le code HTML de la page, on place au fur et à mesure le code JavaScript dont on a besoin en incluant directement dans celui-ci, tout comme pour le HTML, les valeurs des variables du langage hôte.

```
<?PHP
  $Message = '"Hello World!"';
  print '<script type="text/javascript">';
  print '    function confirm(message)
{ ..... }';
  print '</script>';
  print "<a href='#' onClick = 'return confirm(".
$Message.");'>Test</a>";
?>
```

Parmi les avantages, le regroupement de tout le code concernant un point précis. Ici la confirmation du message. Dans le langage hôte, nous avons la définition de la valeur, la génération du code JavaScript qui l'utilise et le code HTML qui permettra de l'invoquer.

Autre avantage, on peut très facilement ne produire que le code JavaScript nécessaire à ce besoin. Si la fonction JavaScript à utiliser dépend d'une propriété dans le langage hôte, un simple `if` permet de produire la bonne fonction.

Bref, je ne vais pas énumérer les avantages et encore moins les inconvénients.

Mon propos est juste de montrer pourquoi cette idée est une fausse bonne idée.
Pour cela revenons, au fonctionnement du Web, c'est-à-dire HTTP.

1. Le client invoque une URL.
2. Le serveur déroule son code et produit dynamiquement une source HTML.
3. Le client reçoit cette source.
    - Il ne peut pas le mettre en cache car il est dynamique et contient des données variables.
    - Il le parse et crée le DOM correspondant.
    - Il donne à l'interpréteur JavaScript la source JavaScript qu'il contient.
4. L'interpréteur JavaScript compile le code (partiellement) et le relie au DOM.
    - L'interpréteur ne peut mettre le code compilé en cache car il est embarqué dans une source qui n'est pas cachée.
5. La page est disponible pour l'utilisateur.
6. L'utilisateur active le code JavaScript par une action (un click dans l'exemple).
7. L'interpréteur JavaScript compile si besoin le code non compilé et l'exécute.
    - Si l'action est encore invoquée, elle est juste exécutée.

Si le client au cours de son activité revient sur la même page, on repart du début.
Le code est régénéré, retransféré, réinterprété par le moteur HTML, recompilé pour le JavaScript et ce à chaque fois qu'on invoque l'URL.

Peut-on gagner en efficacité ? La réponse est oui évidemment, en utilisant le protocole et la capacité du navigateur.

La première chose à faire est de mettre le code JavaScript dans des fichiers séparés

```
function confirm(message) { ..... }
```

et inclure le fichier dans la page.

Reste l'utilisation des variables du langage hôte dans JavaScript.
J'ai donné une solution (Lien 133), avec celle-ci il n'est plus du tout nécessaire de placer les valeurs des variables du langage hôte dans JavaScript.

En quoi une telle solution peut-elle améliorer l'efficacité ?
Reprenons le déroulement des opérations.

1. Le client invoque une URL
2. Le serveur déroule son code et produit dynamiquement une source HTML.
3. Le client reçoit cette source.
    - Il ne peut pas la mettre en cache car elle est dynamique et contient des données variables.
    - Il la parse et crée le DOM correspondant.
    - Il charge les fichiers JavaScript liés.
    - Il met la source JavaScript dans le cache.
4. L'interpréteur JavaScript compile le code (partiellement) et le relie au DOM.
    - L'interpréteur met en cache le code compilé.
5. La page est disponible pour l'utilisateur.
6. L'utilisateur active le code JavaScript par une

action (un clic dans l'exemple).
7. L'interpréteur JavaScript compile si besoin le code non compilé et l'exécute.
    - Si l'action est encore invoquée elle est juste exécutée.

Si la page est rappelée, le fichier JavaScript n'est pas rechargé et il n'est pas recompilé.

Du coup diront certains, on se retrouve avec un gros JavaScript qui contient tous les cas alors qu'en générant le JavaScript à la volée, je peux générer la fonction dont j'ai besoin

```
<?PHP
  $Message = '"Hello World!"';
  print '<script type="text/javascript">';
  print '    function confirm(message) {';
  if ($property == 'some value' {
     print "       //ici le code JavaScript dans un cas";
  } else {
     print "       //ici le code JavaScript dans l'autre cas";
  }
  print '    }';
  print '</script>';
  print "<a href='#' onClick = 'return confirm(".$Message.");'>Test</a>";
?>
```

C'est en fait un faux problème. Il suffit de faire plusieurs fichiers JavaScript et d'inclure le bon.

```
<?PHP
  $Message = '"Hello World!"';
  print '<script type="text/javascript" src="' . $property . '.js">';
  print '</script>';
  print "<a href='#' onClick = 'return confirm" . $property . "(".$Message.");'>Test</a>";
?>
```

On a ainsi le code précis de la fonction que l'on veut dans le JavaScript tout en bénéficiant du cache du compilateur JavaScript et du cache du navigateur.

Cet aspect des choses peut avoir de gros effets.
Sur de tout petits JavaScript, ça ne se voit quasiment pas.
Mais ce n'est pas le cas lorsqu'ils se multiplient ou si le code devient plus gros.

Ajoutez les caches réseau dans l'affaire et la différence peut devenir énorme.

Vous avez sûrement remarqué que certaines applications sur le net sont très lentes à démarrer la première fois et bien plus rapides par la suite.

Imaginez une telle application dont le serveur est à l'autre bout du monde. Tout le code statique, CSS statique, images statiques est mis en cache dans des caches réseau tout au long du parcours (le proxy de l'entreprise fait aussi cela).

Du coup, lorsqu'un autre client vient chercher une de ces ressources et que le réseau la trouve en cache dans l'acheminement de la requête, c'est cette version en cache qui est livrée. Ce n'est donc plus simplement juste entre le client et le serveur qu'on optimise ainsi les transferts mais sur toute la ligne.

Avec JavaScript, il y a en plus le cache du code compilé qui apporte un réel confort à l'utilisateur. L'application devient beaucoup plus fluide et réactive.

Commentez la news de sekaijin en ligne : Lien 134.

Les blogs Développement Web

Le "core" JavaScript s'enrichit de nouvelles méthodes.

C'est une information qui est, il me semble, passée relativement inaperçue mais qui est selon moi assez intéressante à souligner.
Le noyau JavaScript, qui était resté longtemps figé, s'est enrichi avec les dernières versions des navigateurs, de nouvelles méthodes bien utiles.
Pour rappel, le noyau JavaScript (aussi appelé *core* JavaScript), par opposition au DOM JavaScript (ou JavaScript côté client) regroupe les objets natifs de JavaScript et surtout, la partie censée être commune à toutes ses variations.
Il regroupe en particulier les objets natifs Array et String qui ont vu leur prototype amélioré.
Il est à noter que ces ajouts sont aussi disponibles (sauf mention contraire) dans Internet Explorer depuis la version 9.

- L'objet Array

La méthode *every()*
Cette méthode permet d'appliquer à tous les membres du tableau une fonction de rappel afin de savoir si tous les éléments du tableau remplissent une condition.

Syntaxe

```
Array.every(callback, thisObjet);
```

Exemple

```
function isImpair(nb){
  return nb & 1;
}
alert([1,5,17,89].every(isImpair));
alert([1,5,17,89, 100].every(isImpair));
```

La méthode *filter()*
Comme son nom l'indique, cette méthode permet de filtrer les éléments d'un tableau selon le résultat renvoyé par une fonction de rappel.

Syntaxe

```
Array.filter(callback, thisObjet);
```

Exemple

```
function isInferieurADix(nb){
  return nb < 10;
}
alert([2,5,6,8,10,11].filter(isInferieurADix));
```

La méthode *forEach()*
Cette méthode permet d'appliquer un traitement à chaque élément du tableau.

Syntaxe

```
Array.forEach(callback, thisObjet);
```

Exemple

```
var tab = ['a','b','c','d','e'],
  resultat = '',
  i = 0;
function arrayToString(){
  resultat += 'Rang '+i+'\t\tvaleur : '+this[i]
+'\n';
  i++;
}
tab.forEach(arrayToString, tab);
alert(resultat);
```

La méthode *map()*
La méthode *map()* va appliquer à chaque élément du tableau le traitement de la fonction de rappel.**La méthode *map()***

Syntaxe

```
Array.map(callback, thisObjet);
```

Exemple

```
var tab = ['a','b','c','d','e'];
function double(val){
  return val + val;
}
alert(tab.map(double));
```

La méthode *some()*
Similaire à la méthode *every()*, cette méthode va vérifier si au moins un des éléments du tableau est valide selon le résultat renvoyé par la fonction de rappel.

Syntaxe

```
Array.some(callback, thisObjet);
```

Exemple

```
function isInferieurADix(nb){
  return nb < 10;
}
alert([10,20,30,40,50].some(isInferieurADix));
alert([5,10,20,30,40,50].some(isInferieurADix));
```

- L'objet String

Ces méthodes renvoient la nouvelle chaîne mais ne modifient pas celle d'origine.

La méthode *trim()*
Supprime tous les caractères d'espacement en début et fin de chaîne.

Syntaxe

```
String.trim();
```

Exemple

```
var str = '\tTest ';
var strTrimmed = str.trim();
alert('!'+str+'!\n!'+strTrimmed+'!');
```

La méthode *trimRight()*
Supprime les espacements en fin de chaîne.

Syntaxe

```
String.trimRight();
```

Exemple

```
var str = '\tTest ';
var strTrimmed = str.trimRight();
alert('!'+str+'!\n!'+strTrimmed+'!');
```

La méthode *trimLeft()*
Supprime les espacements en début de chaîne.
Attention : étonnamment, cette méthode n'est pas (encore) disponible pour Internet Explorer et Opera...

Syntaxe

```
String.trimLeft();
```

Exemple

```
var str = '\tTest ';
var strTrimmed = str.trimLeft();
alert('!'+str+'!\n!'+strTrimmed+'!');
```

À noter aussi, pour l'objet **Date**, l'apparition de la méthode *toISOString()*.

Retrouvez ce billet blog de Bovino en ligne : Lien 135

Un chat en HTML5 avec les websockets

Voici un tutoriel permettant de créer un chat grâce à l'API websocket en HTML5.

1. Compatibilité

Tous les navigateurs modernes proposent un support du websocket de manière native ou via un plugin.

Voici la liste des navigateurs :

- Chrome : support natif ;
- Safari : support natif ;
- Firefox : support natif ;
- Opéra : support natif mais nécessite de l'activer ;
- Internet Explorer : utilisation d'un prototype des websockets : websockets prototype pour IE 9 (Lien 136).

Pour Firefox

Pour activer les websockets dans Firefox s'ils ne le sont pas (comme dans la version 4 par exemple), il suffit de se rendre dans la barre d'adresse et taper la commande suivante :

```
about:config
```

Une page de confirmation apparaît, continuez. Dans le champ filtre, tapez la recherche suivante :

```
network.websocket
```

Deux options apparaîtront, il faut alors changer leurs valeurs :

- network.websocket.enabled : true
- network.websocket.override-security-block : true

Pour Opera

Dans la barre d'adresse du navigateur, tapez la commande suivante :

```
opera:config#Enable%20WebSockets
```

Cliquez sur la checkbox afin d'activer les websockets, puis cliquez sur "Save".

Dans les deux cas, il faudra redémarrer le navigateur.

2. l'API websocket HTML5

D'aucuns diront que le websocket a une faille de sécurité et qu'il ne faut pas l'utiliser en l'état actuel des choses. C'est vrai, mais il faut se projeter. Les websockets se positionnent comme les remplaçants de l'AJAX. Ils sont rapides, il n'y a que les données qui transitent (très peu de données dans le header de la réponse, voire pas du tout) et le fonctionnement est robuste en termes de charge.

Grâce à l'arrivée du HTML5, nous pouvons donc commencer à utiliser les websockets via l'API websockets.

Celle-ci nous fournit des fonctions nous permettant de mettre très facilement en place une communication via les websockets.

Comment utiliser l'API websockets ?

En JavaScript, il faut d'abord instancier un objet **Websocket** qui prend pour paramètre une URL vers un serveur websocket.

```
var socket = new
Websocket("ws://localhost:11345/serveur.php");
```

Notre connexion est en place, nous allons donc écouter son comportement :

```
socket.onopen = function(e){} /*on "écoute" pour
savoir si la connexion vers le serveur websocket
s'est bien faite */
socket.onmessage = function(e){} /*on récupère
les messages provenant du serveur websocket */
socket.onclose = function(e){} /*on est informé
lors de la fermeture de la connexion vers le
serveur*/
socket.onerror = function(e){} /*on traite les
cas d'erreur*/
```

Pour envoyer un message (des données) vers le serveur websocket, il faut utiliser l'instruction suivante :

```
socket.send('mon message') /*'mon_message' peut
être du JSON mais il conviendra de le stringifier
via JSON.stringify('{"msg": "message type
string"}')*/
```

Enfin, l'API propose de pouvoir clôturer une connexion vers le serveur websocket via l'instruction suivante :

```
socket.close();
```

3. le serveur websocket

Il existe sur le Web de nombreuses solutions pour chaque type de langage :

- JAVA : jWebSocket (Lien 137) ;
- Ruby : web-socket-ruby (Lien 138) ;
- Node JS : Socket.IO-node (Lien 139) ;
- PHP : phpwebsocket (Lien 140).

Il existe également d'autres solutions émergentes telles que :

- kaazing : Lien 141 ;
- wakanda : Lien 142.

Dans notre cas, nous utiliserons *phpwebsocket*. Pour l'utiliser, téléchargez-le et dézippez-le dans le dossier de

votre choix.

Ensuite nous allons modifier légèrement le fichier *server.php* (que j'ai renommé *serveur.php* dans le package à télécharger).

Si toutefois vous rencontrez des problèmes lors de l'installation des fichiers de *phpwebsocket* chez vous, n'hésitez pas à télécharger le package du cours où vous trouverez un serveur websocket tout fait (celui utilisé dans cet article).

```
#!/php -q
<?php  /*  >php -q server.php  */
ini_set('default_socket_timeout', 10);
include_once('WebSocketHandshake.class.php');
error_reporting(E_ALL);
set_time_limit(0);
ob_implicit_flush();
$master  = WebSocket('localhost',11345); /* le
host et le port du serveur websocket */
$sockets = array($master);
$users   = array();
$debug   = true;

/*
 * serveur websocket qui tourne en continu
 */
while(true){
  $changed = $sockets;
  $expect = $sockets;
  socket_select($changed,$write=NULL,
$expect,0,1000000);
  foreach($changed as $socket){
    if($socket==$master){
      $client=socket_accept($master);
      if($client<0){
        console("socket_accept() failed");
        continue;
      }else{
        socket_set_option($client,SOL_SOCKET,
SO_KEEPALIVE, 1) or die('Can not set keepalive');
        connect($client);
      }
    }else{
      $bytes = @socket_recv($socket,
$buffer,2048,0);
      if($bytes==0){ disconnect($socket); }
      else{
        $user = getuserbysocket($socket);
        if(!$user->handshake){
          dohandshake($user,$buffer);
        } else{ process($user,$buffer); }
      }
    }
  }
}

/*
 * fonction pour savoir quel type d'action est
demandé - dans notre cas,
 * il n'y en a qu'un mais s"il y avait plusieurs
actions sur ce serveur
 * en plus du chat, nous aurions d'autres cas
dans le switch
 */
function process($from,$msg){
  console("< ".$from->label." (".$from->userId.")
: \n\t".$msg);
  $msg = unwrap($msg);
  $msg = json_decode($msg,true);
  switch($msg['action']){
    case "ctrl/chat/out"    :
      onCtrlChatOut($from,  $msg['msg']);
      break;
    default : onActionError($from, $msg);
      break;
  }
}
/*
 * réécrit les données sur le socket ouvert
 */
function send($to,$msg){
  say('> '.$to->label.' ('.$to->userId.") :\n\t".
$msg);
  $msg = wrap($msg);
  return socket_write($to->socket,
$msg,strlen($msg));
}
/*
 * envoi du message (des données) à chaque
utilisateur connecté
 * sur le serveur de socket (et sur le même
socket)
 */
function broadcast($msg, $excludeUser=''){
  say(">> ");
  global $users;
  foreach($users as $user){
    send($user, $msg);
  }
}
/*
 *initialisation du socket
 */
function WebSocket($address,$port){
  $master=socket_create(AF_INET, SOCK_STREAM,
SOL_TCP)     or die("socket_create() failed");
  socket_set_option($master, SOL_SOCKET,
SO_REUSEADDR, 1)  or die("socket_option()
failed");
  socket_bind($master, $address, $port)
or die("socket_bind() failed");
  socket_listen($master,20)
or die("socket_listen() failed");
  echo "Server Started : ".date('Y-m-d
H:i:s')."\n";
  echo "Master socket  : ".$master."\n";
  echo "Listening on   : ".$address." port ".
$port."\n\n";
  return $master;
}
/*
 * lors de la connexion d'un utilisateur sur le
serveur,
 * il est enregistré pour le broadcast des
données
 */
function connect($socket){
  global $sockets,$users;
  $user = new User();
  $user->id = uniqid();
  $user->socket = $socket;
  array_push($users,$user);
  array_push($sockets,$socket);
  console($socket." CONNECTED!");
}
/*
 * déconnexion de tous les utilisateurs
 */
```

```
function disconnect($socket){
  global $sockets,$users;
  $found=null;
  $n=count($users);
  for($i=0;$i<$n;$i++){
    if($users[$i]->socket==$socket){
      $found=$i; break;
    }
  }
  if(!is_null($found)){
    array_splice($users,$found,1);
  }
  $index = array_search($socket,$sockets);
  socket_close($socket);
  console($socket." DISCONNECTED!");
  if($index>=0){
    array_splice($sockets,$index,1);
  }
}
/*
 * fonction de la persistance de la connexion
websocket
 */
function dohandshake($user,$buffer){
  console("\nRequesting handshake...");
  console($buffer);
  list($resource,$host,$origin) =
getheaders($buffer);
  console("Handshaking...");
  $handshake = WebSocketHandshake($buffer);
  socket_write($user->socket,
$handshake,strlen($handshake));
  $user->handshake=true;
  console($handshake);
  console("Done handshaking...");
  return true;
}
function getheaders($req){
  $r=$h=$o=null;
  if(preg_match("/GET (.*) HTTP/"   ,$req,
$match)){ $r=$match[1]; }
  if(preg_match("/Host: (.*)\r\n/"  ,$req,
$match)){ $h=$match[1]; }
  if(preg_match("/Origin: (.*)\r\n/",$req,
$match)){ $o=$match[1]; }
  return array($r,$h,$o);
}
/*
 * identification des utilisateurs dans le socket
 */
function getuserbysocket($socket){
  global $users;
  $found=null;
  foreach($users as $user){
    if($user->socket==$socket){
      $found=$user; break;
    }
  }
  return $found;
}
/**
 *      fonction permet de formater le message de
retour
 * à envoyer vers les navigateurs (les
utilisateurs)
 */
function onCtrlChatOut($from, $msg){
  $msg = json_decode($msg);
  if($msg->message==='demo.stop'){
    $msg->message='WebSockets server is going
down...';
    broadcast('{"action": "ws/chat/in",
"msg":'.json_encode($msg).'}');
    die();
  }
  broadcast('{"action": "ws/chat/in",
"msg":'.json_encode($msg).'}');
}
/**
 *              Usefull functions
 */
function     say($msg=""){ echo $msg."\n"; }
function    wrap($msg=""){ return chr(0).
$msg.chr(255); }
function  unwrap($msg=""){ return
substr($msg,1,strlen($msg)-2); }
function console($msg=""){ global $debug;
if($debug){ echo $msg."\n"; } }

class User{
       var $id;
       var $socket;
       var $handshake;
       var $userId=null;   // From GUI
}
?>
```

Le code peut paraître un peu obscur si l'on n'est pas familiarisé avec les sockets en PHP. Ce qu'il faut comprendre c'est que ce script permet d'instancier un serveur de socket et qu'il gère la connexion des utilisateurs ainsi que l'envoi des messages via les sockets.

Bon, nous avons notre serveur websocket mais comment le lancer ?

Il faudra s'assurer que l'extension *php_sockets* est bien chargée dans PHP. Pour ce faire, rendez-vous dans le *php.ini* du PHP que vous utilisez et décommentez la ligne nécessaire.

```
...
;extension=php_soap.dll
extension=php_sockets.dll
;extension=php_sqlite.dll
;extension=php_sqlite3.dll
...
```

Dans notre cas, nous n'avons pas besoin de serveur Web. En effet, nous allons utiliser le serveur websocket. J'entends par là que nous pouvons utiliser notre page de chat en local (sans serveur Web, donc pas de http) et c'est notre serveur websocket qui va s'occuper de faire transiter les données entre les différents clients (navigateurs) connectés au serveur websocket.

Pour cela, il suffit donc de lancer notre serveur websocket en accédant à une console de commande (*cmd* sous Windows par exemple) et accéder au répertoire où se situe notre serveur websocket.

Une fois arrivé au dossier nécessaire, on lance la commande suivante :

```
c:\websocket\serveur\>php -q serveur.php
```

Si le message suivant apparaît, vous avez un serveur de

websocket qui fonctionne et qui n'attend plus que les données :

```
Server Started : 2011-06-30 13:59:03
Master socket  : Resource id #5
Listening on   : localhost port 11345
```

La console ainsi ouverte vous permettra de voir le comportement du serveur sur la connexion d'utilisateurs et de traitement de message (de données).

4. Le chat

Jusqu'ici, nous avons notre serveur qui fonctionne, nous allons maintenant créer l'interface permettant d'envoyer et de recevoir des messages depuis notre chat.

Commençons pour l'IHM que nous allons réaliser en HTML. Voici le code :

```
<!DOCTYPE html>
  <html>
  <head>
    <meta charset="ISO-8859-1">
    <link href="style/style.css" type="text/css"
rel="stylesheet"/>
    <script type="text/javascript"
src="script/script.js"></script>
    <title>Insert title here</title>
  </head>
  <body>
    <div class="sii-chat"> <-- conteneur du chat
-->
      <div>Pseudo : <input type="text" name="sii-
chat-name" /><button class="sii-chat-
login">Valider</button></div> <-- pseudo à saisir
pour le chat -->
      <div class="sii-chat-content"> <-- les
messages apparaitront ici -->
      </div>
      <div>
        <form class="sii-chat-form"
onsubmit="return false;">
          <input type="text" value="" name="sii-
chat-message" disabled="disabled"/><-- saisie du
message à saisir -->
          <button class="sii-chat-send"
disabled="disabled">ok</button> <-- Bouton
d'envoi du message saisi -->
        </form>
      </div>
      <div class="console"></div>
    </div>
    <script type="text/javascript"
src="script/websocket.js"></script>
  </body>
</html>
```

La zone de saisie d'un message et le bouton d'envoi d'un message sont volontairement en "disabled", pour obliger l'utilisateur à renseigner un pseudo qui permettra d'identifier chaque intervenant sur le chat.

Notre IHM est en place, intéressons-nous à la partie JavaScript. Dans un premier temps nous allons mettre en place le websocket via un objet JavaScript. Je crée une classe **WebsocketClass** qui va initialiser la communication avec le serveur et gérer tous les évènements liés au websocket.

La variable **host** permet de donner l'URL du serveur de socket.

```
var WebsocketClass = function(host){
   this.socket = new WebSocket(host);
   this.console =
document.getElementsByClassName('console')[0];
};
```

Dans cette classe, je crée une variable **this.socket** qui récupère l'instanciation de **WebSocket()** ainsi qu'une variable **this.console** pour pouvoir retranscrire les actions du websocket dans ma console affichée sur ma page Web.

Une fois la classe créée, je vais l'étendre grâce au prototypage. Il y aura donc une initialisation du websocket, la gestion des évènements soulevés et une fonction permettant d'envoyer les messages.

Nous allons également placer les variables nécessaires au bon fonctionnement du chat.

Voici le code :

```
var uId = ''; /* pseudo de l'utilisateur*/
var button =
document.getElementsByClassName('sii-chat-send')
[0]; /* bouton d'envoi du message */
var messageInput =
document.getElementsByName('sii-chat-message')
[0]; /* message à envoyer vers le serveur */
var buttonUser =
document.getElementsByClassName('sii-chat-login')
[0]; /* bouton de soumission du pseudo */
var contentMessage =
document.getElementsByClassName('sii-chat-
content')[0]; /* div contenant les messages reçus
par le serveur*/
var WebsocketClass = function(host){
  this.socket = new WebSocket(host);
  this.console =
document.getElementsByClassName('console')[0];
};
WebsocketClass.prototype = {
  initWebsocket : function(){
    var $this = this;
    this.socket.onopen = function(){
      $this.onOpenEvent(this);
    };
    this.socket.onmessage = function(e){
      $this._onMessageEvent(e);
    };
    this.socket.onclose = function(){
      $this._onCloseEvent();
    };
    this.socket.onerror = function(error){
      $this._onErrorEvent(error);
    };
    this.console.innerHTML =
this.console.innerHTML + 'websocket init <br />';
  },
  _onErrorEvent :function(err){
    console.log(err);
    this.console.innerHTML =
this.console.innerHTML + 'websocket error <br
/>';
  },
  onOpenEvent : function(socket){
    console.log('socket opened');
```

```
    this.console.innerHTML =
this.console.innerHTML + 'socket opened Welcome -
status ' + socket.readyState + '<br />';
  },
  _onMessageEvent : function(e){
    e = JSON.parse(e.data);
    if(e.msg.length > 0) e.msg =
JSON.parse(e.msg);
    contentMessage.innerHTML =
contentMessage.innerHTML
     + '><strong>' + e.msg.from + '</strong> : '
+ e.msg.message + '<br />';
    contentMessage.scrollTop =
contentMessage.scrollHeight; /* permet de
scroller automatiquement vers le bas dans la div
contenant la réception des messages */
    this.console.innerHTML =
this.console.innerHTML + 'message event lanched
<br />';
  },
  _onCloseEvent : function(){
    console.log('connection closed');
    this.console.innerHTML =
this.console.innerHTML + 'websocket closed -
server not running<br />';
    uId = '';
    document.getElementsByName('sii-chat-name')
[0].value = '';
    messageInput.disabled = 'disabled';
    button.disabled = 'disabled';
  },
  sendMessage : function(){
    var message = '{"from":"' + uId + '",
"message":"' + messageInput.value + '"}';
    this.socket.send('{"action":"ctrl/chat/out",
"msg":' + JSON.stringify(message) + '}');
    messageInput.value = '';
    this.console.innerHTML =
this.console.innerHTML + 'websocket message send
<br />';
  }
};
```

Notre classe est créée, il ne nous suffit plus que de l'utiliser. Nous en profiterons pour également rajouter des écouteurs d'évènements sur les boutons afin de mettre en place le mécanisme suivant :

l'utilisateur s'identifie et ensuite il aura accès au chat.

```
var socket = new
WebsocketClass('ws://localhost:11345/serveur.php'
); /* on instancie un objet WebsocketClass avec
l'URL en paramètre */
if(button.addEventListener){
  buttonUser.addEventListener('click',
function(e){ /* on écoute l'évènement 'click' sur
le bouton permettant de valider son pseudo */
    e.preventDefault(); /* on stoppe la
propagation */
    socket.initWebsocket(); /* initialisation de
la connexion vers le serveur de socket */
    uId = document.getElementsByName('sii-chat-
name')[0].value; /* récupération de la valeur du
pseudo de l'utilisateur */
    messageInput.disabled = ''; /* on permet
l'accès au chat (aux champs permettant d'envoyer
des messages) */
    button.disabled = '';
    return false; /* on évite le rechargement de
page */
  }, true);
  button.addEventListener('click',function(e){ /*
on écoute l'évènement 'click' sur le bouton
permettant d'envoyer le message */
    e.preventDefault();
    socket.sendMessage(); /* on envoie un message
vers le serveur*/
    return false;
  }, true);
} else{
  console.log('votre navigateur n\'accepte pas le
addevenlistener');
}
```

Ajoutons à l'IHM un brin de CSS pour améliorer le visuel de celle-ci :

```
.sii-chat{
  width:400px;
  padding:10px;
  background:#ccc;
  margin:20px auto;
}
.sii-chat-content{
  min-height:400px;
  max-height:400px;
  overflow:hidden;
  overflow-y:scroll;
  background:#fff;
  box-shadow:0px 0px 5px 0px #000;
  margin-bottom:10px;
}
.console{
  min-height:50px;
  max-height:100px;
  overflow:hidden;
  overflow-y:scroll;
}
.sii-chat-form input[name="sii-chat-message"]{
  width:300px;
  box-shadow:inset 0px 0px 5px 0px #000;
}
.sii-chat-form button{
  width:80px;
  float:right;
  color:#ffffff;
  -moz-box-shadow: 0px 0px 5px #343434;
  -webkit-box-shadow: 0px 0px 5px #343434;
  -o-box-shadow: 0px 0px 5px #343434;
  box-shadow: 0px 0px 5px #343434;
  -moz-border-radius: 5px;
  -webkit-border-radius: 5px;
  border-radius: 5px;
  border: 1px solid #656565;
  filter:
progid:DXImageTransform.Microsoft.gradient(startC
olorstr="34cdf9", endColorstr="3166ff"); /* Pour
IE seulement et mode gradient à linear */
  background: -webkit-gradient(linear, left top,
left bottom, from(#34cdf9), to(#3166ff));
  background: -moz-linear-gradient(top center,
#34cdf9, #3166ff);
}
.sii-chat-form button:active{
  filter:
progid:DXImageTransform.Microsoft.gradient(startC
olorstr="3166ff", endColorstr="34cdf9"); /* Pour
IE seulement et mode gradient à linear */
  background: -webkit-gradient(linear, left top,
left bottom, from(#3166ff), to(#34cdf9));
  background: -moz-linear-gradient(top center,
#3166ff, #34cdf9);
}
```

Et voilà, le chat est terminé et fonctionnel. Pour bien tester la démonstration, ouvrez deux navigateurs avec la page de chat et conversez entre les deux. Vous verrez que l'un et l'autre se mettent à jour en même temps et rapidement.

5. Package

Télécharger le package : Lien 143.

Retrouvez l'article de Jerome Debray en ligne : Lien 144.

Introduction à l'API Google Maps

L'API Google Maps fournit une interface intuitive et très réactive construite en utilisant les technologies AJAX. C'est une API ouverte permettant la personnalisation de la carte y compris la possibilité d'ajouter au sein de l'application des données spécifiques à la carte (personnalisation des contrôles, gestion des événements, création des marqueurs avec infobulle...). Encore mieux, Google donne accès à ce service gratuitement !
Dans cet article nous allons examiner quelques-unes des fonctionnalités de base fournies par l'API Google Maps.

1. Qu'est-ce qu'une API (Application Programming Interface) ?

Une API (Lien 145) est une interface fournie par un programme informatique. Elle permet l'interaction des programmes les uns avec les autres.

D'un point de vue technique c'est un ensemble de fonctions, procédures ou classes mises à disposition par une bibliothèque logicielle, un système d'exploitation ou un service.

Toute application Web peut autoriser ou non des développeurs tiers à utiliser une partie de ses fonctionnalités et ce de plusieurs façons :

- en téléchargeant une bibliothèque de fonctions pour l'inclure directement sur son site ;
- en récupérant une donnée précise en construisant une URL comme avec Google charts : https://chart.googleapis.com/chart?cht=chart_type&chd=chart_data&chs=cha... ;
- comme Google Maps, en incluant directement la bibliothèque en ligne.

2. Installation de l'API

On va inclure la bibliothèque Google Maps directement dans le code JavaScript de la page. Cette URL permet d'accéder à l'ensemble des fonctionnalités de l'API.

Exemple d'ajout d'une balise script pour inclure les fonctions de Google Maps dans l'application cible :

```
<script type="text/javascript"
src="http://maps.google.com/maps/api/js"></script
>
```

3. Paramètres de l'URL

Plusieurs déclinaisons de l'API sont disponibles en ajoutant des paramètres à l'URL de base ; en voici quelques exemples :

- **sensor** (true ou false) : utilisation ou non de la géolocalisation ;
- **language** : langue des textes à afficher sur la carte ;
- **region** : le pays.

On peut ajouter d'autres paramètres à l'URL pour inclure des librairies additionnelles :

- **geometry** : inclut des fonctions nécessaires pour le calcul scalaire de valeurs géométriques sur la surface de la Terre ;
- **adsense** : permet à votre application Maps d'inclure des annonces contextuelles, vous permettant de partager les revenus publicitaires pour les annonces diffusées aux utilisateurs ;
- **panoramio** :permet d'ajouter la fonctionnalité Panoramio : quand on clique sur l'icône d'une photo avec Panoramio, on a par défaut une pop-up qui s'ouvre avec des informations et la photo en plus grand.

On peut ajouter plusieurs librairies Google Maps dans l'URL, il suffit de les séparer par des virgules.

```
<script type="text/javascript"
src="http://maps.google.com/maps/api/js?
libraries=adsense,geometry&sensor=true"></script>
```

On se retrouve donc avec une déclaration finale de ce type :

```
<head>
<meta name="viewport" content="initial-scale=1.0,
user-scalable=no" />
<script type="text/javascript"
src="http://maps.google.com/maps/api/js?
sensor=true"></script>
<style type="text/css">
html { height: 100% }
body { height: 100%; margin: 0px; padding: 0px }
#map_canvas { height: 100% }
</style>
</head>
```

On spécifie une taille pour les balises html et body (pour les navigateurs plus anciens il faut préciser la taille des éléments parents sinon ceux-ci supposent que la taille est de 0x0px).
Ici la balise meta spécifie le mode plein écran et que la carte n'est pas redimensionnable par l'utilisateur.

4. Google Maps avec SSL

L'utilisation du SSL (Lien 146) avec Google Maps permet d'éviter les avertissements de sécurité dans la plupart des navigateurs.
Ce genre d'utilisation est prévu pour les sites collectant des données confidentielles sur des utilisateurs (telles que la localisation personnelle ou professionnelle d'un utilisateur dans des requêtes).

Exemple d'implémentation :

```
<script type="text/javascript" src="https://maps-api-ssl.google.com/maps/api/js?v=3.4&sensor=true"></script>
```

5. Mise en place de la carte

Nous allons afficher la carte une fois la page HTML chargée. Pour cela nous allons ajouter l'attribut onload à la balise body avec en paramètre la fonction d'initialisation de la carte :

```
<body onload="initialize()">
<script type="text/javascript">
   function initialize() {
       //...
   }
</script>
```

Ensuite il reste à instancier un nouvel objet google.maps.Map comme ceci :

```
function initialize() {
   map = new google.maps.Map(document.getElementById("map_canvas"), {
       zoom: 19,
       center: new google.maps.LatLng(48.8695490, 2.3513734),
       mapTypeId: google.maps.MapTypeId.ROADMAP
   });
}
```

La nouvelle carte sera contenue dans la div "*map_canvas*", centrée sur les coordonnées de latitude 48.8695490 et longitude 2.3513734 (qui représentent la position de The Coding Machine à Paris 2e) et avec un niveau de zoom très précis.

Le paramètre mapTypeId sert à définir le type de carte que l'on souhaite, il en existe quatre :

- **ROADMAP** : affiche le plan classique, sans image satellite ni relief ;
- **SATELLITE** : pour les photos satellite ;
- **HYBRID** : pour afficher les photos satellite avec le plan superposé (les routes, le nom des villes) ;
- **TERRAIN** : affiche les différences de reliefs (montagnes, rivières, etc.).

6. Chargement asynchrone de l'API Google Maps

Généralement, l'API est chargée au démarrage de l'application. Cela peut occasionner un ralentissement du temps d'affichage de la page. Il est possible de charger l'API Google Maps en mode asynchrone, c'est-à-dire en différé ; on utilise pour cela le paramètre callback dans l'URL qui appellera la fonction d'initialisation de la carte.

```
function initialisation(){
   var maLatlng = new google.maps.LatLng(48.8695490, 2.3513734);
   var mesOptions = {
       zoom: 8,
       center: maLatlng,
       mapTypeId: google.maps.MapTypeId.ROADMAP
   }
   var carte = new google.maps.Map(document.getElementById("map_canvas"), mesOptions);
}
// Création de la balise script pour inclure les fonctions de Google Maps dans notre application
function loadScript() {
   var script = document.createElement("script");
   script.type = "text/javascript";
   script.src = "http://maps.google.com/maps/api/js?sensor=false&callback=initialisation";
   document.body.appendChild(script);
}
// Au chargement de la page on appelle la fonction loadScript qui appelle la fonction initialisation() grâce au paramètre callback
window.onload = loadScript;
```

7. Gestion des événements Google Maps

Il est possible de capturer un événement sur un marqueur ou sur la carte grâce à la méthode addListener(). Il existe différents types d'événements simples :

- 'click' ;
- 'dblclick' ;
- 'mouseup' ;
- 'mousedown' ;
- 'mouseover' ;
- 'mouseout' ;
- zoom_changed.

Ces événements ressemblent à des événements DOM standard, mais ils font en réalité partie intégrante de l'API Google Maps.

```
var carte;
function initialisation(){
    // On rentre les coordonnées(latitude, longitude) de notre choix dans une variable
    var maLatlng = new google.maps.LatLng(-25.363882,131.044922);

    // On établit les options de notre choix :
    // la profondeur du zoom, les coordonnées sur lesquelles la carte sera centrée, le type de vue (satellite, plan...)
    var mesOptions = {
       zoom: 4,
       center: maLatlng,
       mapTypeId: google.maps.MapTypeId.ROADMAP
    }

    // On crée notre carte en lui passant toutes nos options en paramètre
    carte = new google.maps.Map(document.getElementById("map_canvas"), mesOptions);

    // On place un listener sur la carte qui contrôle une action qui sera déclenchée lors de l'événement 'zoom_changed'
    // Quand le zoom sera modifié la carte sera recentrée sur les coordonnées de The Coding Machine
    google.maps.event.addListener(carte,'zoom_changed', function() {
       setTimeout(allerChezTCM,3000);
    });
```

```
    // On crée un marqueur que l'on positionne grâce au paramètre "position"
    var marker = new google.maps.Marker ({
       position: maLatlng,
       map: carte,
       title: "Hello world :) !"
    });

    // On place un listener sur le marqueur qui contrôle une action qui sera déclenchée lors de l'évènement 'click'
    // Quand on clique sur le marqueur, le zoom de la carte passera à 8
    google.maps.event.addListener(marker, 'click', function(){
       carte.setZoom(8);
    });
}
function allerChezTCM() {
   var tcm = new google.maps.Latlng(48.8695490, 2.3513734);
   map.setCenter(tcm);
}
```

8. Les contrôles Google Maps

Il est possible d'activer ou de désactiver un contrôle en changeant la valeur de sa propriété à true ou false :

```
{
   panControl: boolean,
   zoomControl: boolean,
   mapTypeControl: boolean,
   scaleControl: boolean,
   streetViewControl: boolean,
   overviewMapControl: boolean
}
```

On peut également modifier les options d'un contrôle (position et style)

```
zoomControl: true,
zoomControlOptions: {
   style: google.maps.ZoomControlStyle.LARGE,
   position: google.maps.ControlPosition.LEFT_CENTER
},
```

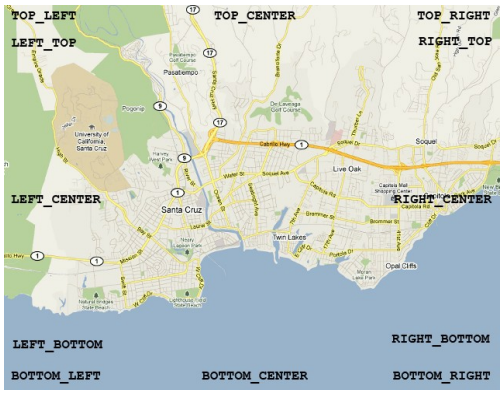
En voici quelques exemples :

- ZoomControlStyle.LARGE : zoom standard avec le slider ;
- ZoomControlStyle.SMALL : affiche un mini zoom avec juste les icônes + et - ;
- ZoomControlStyle.DEFAULT : choisit le bon format de zoom en fonction de taille de la carte du type d'appareil (mobile ou non) ;
- MapTypeControlStyle.HORIZONTAL_BAR : affiche une barre horizontale pour la sélection du type d'affichage de la carte ;
- MapTypeControlStyle.DROPDOWN_MENU : permet de choisir le type de carte (plan, satellite, terrain, hybride) via une liste déroulante ;
- MapTypeControlStyle.DEFAULT : comportement par défaut, dépend de la taille de l'écran.

Exemple de configuration d'un contrôle :

```
//On établit une liste déroulante pour le contrôle MapType et on spécifie que le Zoom control utilise un minizoom
function initialize() {
   var myOptions = {
       zoom: 4,
       center: new google.maps.LatLng(-33, 151),
       mapTypeControl: true,
       mapTypeControlOptions: {
           style: google.maps.MapTypeControlStyle.DROPDOWN_MENU
       },
       zoomControl: true,
       zoomControlOptions: {
           style: google.maps.ZoomControlStyle.SMALL
       },
       mapTypeId: google.maps.MapTypeId.ROADMAP
   }
   var map = new google.maps.Map(document.getElementById("map_canvas"), myOptions);
}
```

Voici toutes les positions possibles d'un contrôle sur une carte :



9. Les marqueurs Google Maps

Voici comment créer un marqueur simple avec un contenu info bulle HTML :

```
function initialize() {
   var maLatlng = new google.maps.LatLng(48.8695490, 2.3513734);
   var mesOptions = {
       zoom: 4,
       center: myLatlng,
       mapTypeId: google.maps.MapTypeId.ROADMAP
   }
   var carte = new google.maps.Map(document.getElementById("map_canvas"), mesOptions);
   var image = 'beachflag.png';
   // Instanciation de notre marqueur
   var marker = new google.maps.Marker({
       position: maLatLng,
       map: carte,
       icon: image
   });
   var message = "Vous êtes ici !";
   var infowindow = new google.maps.InfoWindow({
       content: message,
       size: new google.maps.Size(50,50)
   });
```

```
   google.maps.event.addListener(marker, 'click',
function() {
       infowindow.open(carte,marker);
   });
}
```

Les marqueurs sont personnalisables à l'aide de la propriété icon, il suffit de lui passer une image en paramètre.

En bonus, il est possible d'ajouter une animation aux marqueurs en appelant la méthode setAnimation() sur l'objet marker. Deux types sont possibles :

- **DROP** : indique que ce marqueur devrait tomber du haut de la carte jusqu'à son emplacement définitif quand il est affiché sur la carte la première fois. L'animation cesse une fois que le curseur est en place. Ce type d'animation est généralement spécifié lors de la création du marqueur ;
- **BOUNCE** : indique que le marqueur doit remuer. Un marqueur de ce type va continuer de bouger jusqu'à ce que sa propriété soit explicitement désactivée (= null).

```
var stockholm = new google.maps.LatLng(59.32522,
18.07002);
var tcm = new google.maps.LatLng(48.8695490,
2.3513734);
var marker;
var map;
function initialize() {
   var mapOptions = {
       zoom: 13,
       mapTypeId: google.maps.MapTypeId.ROADMAP,
       center: stockholm
   };
   map = new
google.maps.Map(document.getElementById("map_canv
as"), mapOptions);
   marker = new google.maps.Marker({
       map:map,
       draggable:true,
       animation: google.maps.Animation.DROP,
       position: tcm
   });
   google.maps.event.addListener(marker, 'click',
toggleBounce);
}

function toggleBounce() {
   if (marker.getAnimation() != null) {
       marker.setAnimation(null);
   } else {
       marker.setAnimation(google.maps.Animation.
BOUNCE);
   }
}
```

10. Utiliser la géolocalisation

```
// On teste tout d'abord si le navigateur prend
en charge la géolocalisation HTML5
if (navigator.geolocation){
   var watchId =
navigator.geolocation.watchPosition(
       successCallback,
       null,
       {enableHighAccuracy:true}
   );
}
else{
   alert("Votre navigateur n'est pas compatible
avec la géolocalisation HTML 5");
}
function successCallback(position){
   map.panTo(new
google.maps.LatLng(position.coords.latitude,
position.coords.longitude));
   var marker = new google.maps.Marker({
       position: new
google.maps.LatLng(position.coords.latitude,
position.coords.longitude),
       map: map
   });
}
```

À chaque déplacement, un nouveau marqueur est placé ; dans le cas d'une perte de signal le marqueur se positionne au relais 3G le plus proche.

Pour contourner le problème on utilise la propriété accuracy :

```
if(position.coords.accuracy < 100) {
   //...
}
```

11. Utilisation de la librairie Geometry pour le calcul de distance

Tout ce que vous avez à faire est d'inclure la librairie *Geometry* lors de l'intégration de l'API à l'application, puis d'appeler la méthode computeDistanceBetween() et lui passer deux objets LatLng en paramètres :

```
var nyc = new google.maps.LatLng(40.715,
-74.002);
var london = new google.maps.LatLng(51.506,
-0.119);
var distance =
google.maps.geometry.spherical.computeDistanceBet
ween(nyc, london);
```

Retrouvez l'article de The Coding Machine en ligne : Lien 147

Liens

Lien 01 : http://eclipse.developpez.com/cours/?page=platform-cat#plugin-dev
Lien 02 : http://www.eclipse.org/downloads
Lien 03 : http://maven.apache.org/
Lien 04 : http://mbaron.developpez.com/eclipse/introplugin/
Lien 05 : http://keulkeul.blogspot.com/search/label/Tycho
Lien 06 : http://www.eclipse.org/tycho/
Lien 07 : ftp://ftp-developpez.com/mbaron/eclipse/tycho.zip
Lien 08 : http://mbaron.ftp-developpez.com/eclipse/tycho.zip
Lien 09 : http://mbaron.developpez.com/eclipse/introtycho/
Lien 10 : http://www.developpez.net/forums/d998128/systemes/autres-systemes/mobiles/android-detrone-liphone-aux-usa-termes-dabonnes-premiere/
Lien 11 : http://www.developpez.com/actu/38093/Android-Ice-Cream-Sandwich-lance-officiellement-avec-le-telephone-Galaxy-Nexus/
Lien 12 : http://www.developpez.com/actu/28153/Google-devoile-enfin-Android-3-0-Honeycomb-dans-lequel-tout-est-pense-pour-les-tablettes/
Lien 13 : http://linux.developpez.com/actu/30188/Google-retarde-la-diffusion-du-code-source-d-Android-3-une-demarche-qui-commence-a-faire-debat-dans-la-communaute-open-source/
Lien 14 : http://www.developpez.com/actu/27224/Developper-pour-Android-sans-Java-bientot-possible-en-C-grace-a-la-5eme-revision-du-Native-Development-Kit/
Lien 15 : http://source.android.com/source/downloading.html
Lien 16 : http://developer.android.com/sdk/ndk/index.html
Lien 17 : http://www.developpez.net/forums/d1152665/java/general-java/java-mobiles/android/google-publie-code-source-dandroid-4-kit-developpement-natif-los-supporte-nouvelles-api/
Lien 18 : http://code.google.com/p/ksoap2-android/downloads/detail?name=ksoap2-android-assembly-2.4-jar-with-dependencies.jar
Lien 19 : http://michel-dirix.developpez.com/tutoriels/android/android-et-webservices/
Lien 20 : http://java.developpez.com/cours/
Lien 21 : http://jmdoudoux.developpez.com/cours/developpons/java/chap-presentation.php#presentation-6
Lien 22 : http://java.developpez.com/outils/edi/?page=advanced#eclipse
Lien 23 : http://java.developpez.com/outils/edi/?page=advanced
Lien 24 : http://eclipse.developpez.com/cours/
Lien 25 : http://javasearch.developpez.com/j2se/1.6.0/docs/api/java/awt/LayoutManager2.html
Lien 26 : http://javasearch.developpez.com/j2se/1.6.0/docs/api/java/util/EventListener.html
Lien 27 : http://java.developpez.com/cours/?page=desktop-cat
Lien 28 : http://gfx.developpez.com/tutoriel/java/swt/
Lien 29 : http://eclipse.developpez.com/cours/?page=platform-cat#plugin-dev
Lien 30 : http://jmdoudoux.developpez.com/cours/developpons/java/chap-ria-rda.php
Lien 31 : http://javasearch.developpez.com/j2se/1.6.0/docs/api/java/util/ResourceBundle.html
Lien 32 : http://jmdoudoux.developpez.com/cours/developpons/java/chap-i18n.php
Lien 33 : http://jmdoudoux.developpez.com/cours/developpons/java/index.php
Lien 34 : http://hikage.developpez.com/java/tutoriel/spring/i18n/internationalisation/base-donnees/
Lien 35 : http://javasearch.developpez.com/j2se/1.6.0/docs/api/java/util/PropertyResourceBundle.html
Lien 36 : http://uml.developpez.com/cours/
Lien 37 : http://abrillant.developpez.com/tutoriel/java/design/pattern/introduction/
Lien 38 : http://ego.developpez.com/spring/
Lien 39 : http://adiguba.developpez.com/tutoriels/java/tiger/annotations/
Lien 40 : http://smeric.developpez.com/java/astuces/tests/
Lien 41 : http://rpouiller.developpez.com/tutoriels/java/tests-unitaires-junit4/
Lien 42 : http://spalud.developpez.com/tutoriel/java/testng/
Lien 43 : http://baptiste-wicht.developpez.com/tutoriels/java/tests/mocks/easymock/
Lien 44 : http://fchabanois.developpez.com/tutorial/java/jmockit/
Lien 45 : http://dboissier.developpez.com/tutoriels/test-driven-development/?page=authentification#LIII-B-6
Lien 46 : http://beuss.developpez.com/tutoriels/java/jakarta/commons/logging/
Lien 47 : http://beuss.developpez.com/tutoriels/java/jakarta/log4j/
Lien 48 : http://baptiste-wicht.developpez.com/tutoriels/java/slf4j/
Lien 49 : http://www.insideit.fr/post/2009/11/23/SLF4J-LOGBack
Lien 50 : http://jguillard.developpez.com/JDBC/
Lien 51 : http://jmdoudoux.developpez.com/cours/developpons/java/chap-jpa.php
Lien 52 : http://ippon.developpez.com/articles/java/persistance/solutions/
Lien 53 : http://ippon.developpez.com/articles/java/persistance/ibatis/
Lien 54 : http://java.developpez.com/cours/?page=outils-cat#maven
Lien 55 : http://linsolas.developpez.com/articles/java/qualite/sonar/
Lien 56 : http://linsolas.developpez.com/articles/hudson/
Lien 57 : http://loic-mathieu.developpez.com/conception/article/cruise-control/
Lien 58 : http://java.developpez.com/cours/?page=outils-cat#ant
Lien 59 : http://linsolas.developpez.com/articles/java/outils/builds/
Lien 60 : http://tahe.developpez.com/java/baseswebmvc/
Lien 61 : http://mbaron.developpez.com/javaee/tomcat/
Lien 62 : http://ego.developpez.com/spring/
Lien 63 : http://valtech.developpez.com/articles/java/javaee/jboss/seam/
Lien 64 : http://mbaron.developpez.com/javaee/jsf/
Lien 65 : http://jmdoudoux.developpez.com/cours/developpons/java/chap-ejb.php
Lien 66 : http://x-plode.developpez.com/tutoriels/netbeans/nouvelles-fonctionnalites-dans-jca-1-6/
Lien 67 : http://aldian.developpez.com/cours/introduction-a-l-ecosysteme-java/
Lien 68 : http://argyronet.developpez.com/office/access/setformtoright/
Lien 69 : http://www.developpez.net/forums/d724265/logiciels/microsoft-office/access/contribuez/positionner-formulaire-sous-controle/#post4201158
Lien 70 : http://claudeleloup.developpez.com/tutoriels/access/positionner-formulaire/PositionnerFormulaire.mdb
Lien 71 : http://claudeleloup.developpez.com/tutoriels/access/positionner-formulaire/
Lien 72 : http://fauconnier.developpez.com/excel/bases/references
Lien 73 : http://fauconnier.developpez.com/videos/excel/excel-comparaison-listes-mise-en-forme-conditionnelle.swf
Lien 74 : http://fauconnier.developpez.com/tutoriels/excel/video-excel-2010-comparer-listes-grace-mise-forme-conditionnelle/

Lien 75 : http://datametric.developpez.com/tutoriels/sas/import_export_sas92_x64/?page=page_1#L2-C-1
Lien 76 : http://datametric.developpez.com/tutoriels/sas/import_export_sas92_x64/?page=page_1#L2-C-2
Lien 77 : http://datametric.developpez.com/tutoriels/sas/import_export_sas92_x64/?page=page_1#L2-C-3
Lien 78 : http://datametric.developpez.com/tutoriels/sas/import_export_sas92_x64/?page=page_1#L2-D-1
Lien 79 : http://datametric.developpez.com/tutoriels/sas/import_export_sas92_x64/?page=page_1#L2-D-2
Lien 80 : http://datametric.developpez.com/tutoriels/sas/import_export_sas92_x64/?page=page_1#L2-D-3
Lien 81 : http://support.sas.com/kb/40/383.html
Lien 82 : http://support.sas.com/kb/14/647.html
Lien 83 : http://en.wikipedia.org/wiki/Microsoft_Jet_Database_Engine
Lien 84 : http://support.sas.com/documentation/cdl/en/acpcref/63184/HTML/default/viewer.htm
Lien 85 : http://support.sas.com/documentation/cdl/en/acpcref/63184/HTML/default/viewer.htm
Lien 86 : http://support.sas.com/kb/41/060.html
Lien 87 : http://support.sas.com/kb/41/060.html
Lien 88 : http://support.sas.com/kb/33/228.html
Lien 89 : http://support.sas.com/kb/40/383.html
Lien 90 : http://support.sas.com/kb/40/409.html
Lien 91 : http://support.sas.com/kb/37/479.html
Lien 92 : http://support.sas.com/kb/40/597.html
Lien 93 : http://support.sas.com/kb/36/967.html
Lien 94 : http://datametric.developpez.com/tutoriels/sas/import_export_sas92_x64/
Lien 95 : http://qt.developpez.com/actu/40074/Sortie-de-Qt-4-8-le-framework-C-ameliore-son-architecture-pour-faciliter-le-portage-vers-de-nouvelles-plateformes/
Lien 96 : http://qt-devnet.developpez.com/tutoriels/python/pyside/installer/
Lien 97 : http://developer.qt.nokia.com/wiki/Category:LanguageBindings::PySide::Downloads
Lien 98 : http://pyqt.developpez.com/actu/36457/PySide-le-binding-Python-de-Qt-en-danger-suite-a-l-arret-des-subsides-de-Nokia/
Lien 99 : http://www.developpez.net/forums/d1171611/autres-langages/python-zope/gui/pyside-pyqt/sortie-pyside-1-1-0-a/
Lien 100 : http://qt.developpez.com/actu/38218/Le-Qt-Project-est-la-le-projet-d-open-gouvernance-pour-le-framework-C-est-arrive-a-terme/
Lien 101 : http://mesa3d.org/
Lien 102 : http://www.developpez.net/forums/d1145082/c-cpp/bibliotheques/qt/ouverture-qt-dev-days-2011-munich/#post6311498
Lien 103 : http://www.developpez.net/forums/d1144709/c-cpp/bibliotheques/qt/vivats-qt-project/
Lien 104 : http://www.developpez.net/forums/d1145082/c-cpp/bibliotheques/qt/reportage-direct-deuxieme-jour-qt-dev-days-2011-munich/#post6311702
Lien 105 : http://www.developpez.net/forums/d1145082/c-cpp/bibliotheques/qt/reportage-direct-deuxieme-jour-qt-dev-days-2011-munich/
Lien 106 : http://qt.developpez.com/doc/4.7/qpainter
Lien 107 : http://qt.developpez.com/doc/4.7/vue-graphique-graphics-view/
Lien 108 : http://qt.developpez.com/doc/4.7/QtOpenGL/
Lien 109 : http://qt.developpez.com/doc/4.7/Qtmultimedia/
Lien 110 : http://qt.developpez.com/doc/4.7/qtsvg/
Lien 111 : http://qt.developpez.com/doc/4.7/phonon-overview/
Lien 112 : http://qt.developpez.com/doc/4.7/model-view-programming/
Lien 113 : http://qt.developpez.com/doc/4.7/exemples/
Lien 114 : http://qt.developpez.com/doc/4.7/qt4-scribe/
Lien 115 : http://qt.developpez.com/doc/4.7/richtext/
Lien 116 : http://qt.developpez.com/doc/4.7/qtextdocumentwriter/
Lien 117 : http://qt.developpez.com/doc/4.7/qtwebkit/
Lien 118 : http://qt.developpez.com/doc/4.7/demos-browser/
Lien 119 : http://qt.developpez.com/tutoriels/integrer-gadgets-logiciels-webkit/
Lien 120 : http://qt-devnet.developpez.com/tutoriels/qt/livre-blanc/
Lien 121 : http://vincent-vande-vyvre.developpez.com/tutoriels/pyqt/manipulation-images/
Lien 122 : http://www.riverbankcomputing.co.uk/static/Docs/PyQt4/html/qgraphicsitem.html#GraphicsItemFlag-enum
Lien 123 : http://www.pyside.org/docs/pyside/PySide/QtGui/QGraphicsItem.html#PySide.QtGui.PySide.QtGui.QGraphicsItem.GraphicsItemFlag
Lien 124 : http://vincent-vande-vyvre.developpez.com/tutoriels/pyqt/qgraphicsitem/
Lien 125 : http://www.nih.at/libzip/
Lien 126 : http://www.nih.at/libzip/libzip.html
Lien 127 : http://slash.developpez.com/tutoriels/c/utilisation-libzip/#L2-E-2
Lien 128 : http://slash.ftp-developpez.com/tutoriels/c/utilisation-libzip/fichiers/zipzap.c
Lien 129 : http://slash.developpez.com/tutoriels/c/utilisation-libzip/
Lien 130 : http://vb.developpez.com/telecharger/detail/id/1238/MSDN-VB6-Fr
Lien 131 : http://vb.developpez.com/telecharger/detail/id/1249/MZ-Tools
Lien 132 : http://darkvader.developpez.com/tutoriels/vb/debogage-visual-basic-6/
Lien 133 : http://www.developpez.net/forums/d1075435/webmasters-developpement-web/contribuez/php-javascript-methode/#post6338448
Lien 134 : http://www.developpez.net/forums/d1151380/webmasters-developpement-web/javascript/publications-javascript-ajax/pourquoi-generer-code-javascript-fausse-bonne-idee/
Lien 135 : http://blog.developpez.com/web/p10401/web/javascript/le-core-javascript-s-enrichit-de-nouvell/
Lien 136 : http://html5labs.interoperabilitybridges.com/html5labs/prototypes/websockets/websockets/info/
Lien 137 : http://jwebsocket.org/
Lien 138 : https://github.com/gimite/web-socket-ruby
Lien 139 : https://github.com/LearnBoost/Socket.IO-node
Lien 140 : http://code.google.com/p/phpwebsocket/
Lien 141 : http://www.kaazing.com/download
Lien 142 : http://www.wakanda.org/download/
Lien 143 : http://sii-rennes.developpez.com/telechargements/websockets.zip
Lien 144 : http://sii-rennes.developpez.com/articles/un-chat-en-html5-avec-les-websockets/
Lien 145 : http://dico.developpez.com/html/1453-Langages-API-Application-Programming-Interface.php
Lien 146 : http://dico.developpez.com/html/1589-Securite-SSL-Secure-Sockets-Layer.php
Lien 147 : http://thecodingmachine.developpez.com/tutoriels/javascript/introduction-api-google-maps/