Developpez Magazine

Edition de Novembre Janvier 2006/2007.
Numéro 7.
Magazine en ligne gratuit.
Diffusion de copies conformes à l'original autorisée.
Réalisation : Baptiste Wicht et Sylvain Luce
Rédaction : la rédaction de Developpez
Contact : magazine@redaction-developpez.com

Index

Editorial

La rentrée est passée et le magazine de Developpez.com revient avec son 7ème numéro.

N'hésitez pas à passer sur les forums pour donner votre avis ou faire des suggestion pour cette publication

La rédaction

Tutoriel JAVA

Présentation de Java SE 6

La version 6 du langage Java est sortie, cet article va vous en présenter toutes les nouveautés.

par **Frédéric Martini**

Page 5

Article PHP

L'upload des fichiers

Vous avez envie de permettre aux visiteurs de votre site d'uploader des fichiers sur votre serveur ? Mais vous ne savez pas comment faire ? Alors ce tutoriel est fait pour vous. Vous verrez, en lisant ce tutoriel qu'uploader un fichier en php est faisable mais qu'en plus c'est très simple ! ;o)

par **Antoine Herault**

Page 11

Affichage Bi-écran avec X.org sous Linux

1. Avant-Propos

Il y a quelques années déjà, ma possibilité de migrer sous Linux était suspendue au fait de pouvoir avoir deux écrans : un pour programmer, l'autre pour tester la page Internet générée avec le code sous les yeux, Mandrake 8.0 n'était pas encore sortie...

Avec l'aide de Mandrake 8.X, j'y suis arrivé, depuis je suis sous Linux et j'aime ça !

Quand le boulot m'a donné un portable sous Windows, il va de soit que j'ai aussitôt voulu y faire tourner Linux avec deux écrans. J'y suis arrivé toujours avec Mandriva (2005 à l'époque) et avec des documents trouvés sur Internet. Les fichiers de configuration qui fonctionnaient sous Mandriva m'ont servi à faire du double écran sur ce même PC avec Ubuntu (5.10 et 6.06) ainsi qu'avec Debian sid.

Je vais commencer à vous faire partager mes fichiers de configuration avec deux cartes écran puis celui de mon portable.

1. Avec deux cartes écrans différentes

Vous avez récupéré une vieille carte écran PCI et un écran inutilisé.

Vous avez un peu de place sur votre bureau et un slot PCI libre au fond de votre PC.

Vous avez décidé de profiter de tout ça pour mettre deux écrans sur vote PC, allez-y. J'espère vous guider !

Attention : les très vieilles cartes ISA ne sont pas concernées par cette manipulation, elles ne supportent pas l'affichage en duo (Linux et Windows => je pense que c'est un problème matériel).

1.1. Exemple de xorg.conf

C'est un exemple tiré d'une mandriva 2006, mais mon expérience m'a montré qu'on peut récupérer la partie affichage sur Debian ou Ubuntu.

1.2. Section spécifique

Cette section, je vous conseille de la laisser générer par votre distribution. Pour une Mandriva 2006 ou précédente :

```
FontPath "/usr/share/fonts/misc:unscaled"
```

Pour une Mandriva 2007 :

```
FontPath "/usr/X11R6/lib/X11/fonts/75dpi:unscaled"

FontPath "/usr/X11R6/lib/X11/fonts/TTF"
```

Pour une Debian sid en 2006 :

```
FontPath "/usr/share/fonts/X11/75dpi"
FontPath "/usr/lib/X11/fonts/75dpi"
```

Ajouter une police de caractères m'a déjà permis de sauver une mise à jour plantée qui empêchait de lancer X car la font fixed manquait. J'ai rajouté la police TTF et le serveur X a enfin pu repartir.

La section qui semble vraiment dépendante de la distribution est la section module.

1.3. La gestion de l'affichage

Cette partie informe sur les cartes écrans (section Device), les écrans (section Monitor), les couples carte/écran (section Screen) et comment on met tout ça ensemble avec clavier + souris définis dans la première partie (section ServerLayout). Comme elle est indépendente de la distribution, vous pouvez la faire générer par Mandriva et la conserver sur votre distribution.

La section la plus critique est la section Device : elle doit définir le driver utilisé (Driver) pour la carte ainsi que son emplacement dans le fond de panier (BusID). Un changement de place de la carte PCI changera son BusID, donc la configuration de cette partie. Toujours sur les cartes écrans, j'ai deux remarques à faire. La première, c'est que je ne suis pas arrivé à faire en tourner deux avec les mêmes drivers (je n'ai guère essayé car j'ai toujours eu des cartes cartes ISA en stock) et si j'avais une nVidia, en prenant le noyau nvidia de Mandrake (pas testé sous Mandriva), je ne pouvais plus faire fonctionner l'autre (problème de configuration noyau sûrement, mais pas testé).

La section que l'on doit configurer soi même, c'est ServerLayout : je n'ai pas trouvé de logiciel pour faire le placement relatif des écrans. J'ai bien essayé de faire un programme copiant un peu l'interface de Windows, mais je me suis arrêté faute de tests suffisants. Cette section comporte aussi l'option xinerama, pour qu'elle soit valide, il faut que la profondeur (nombre de couleurs) soit la même sur les écrans. Elle se règle dans la section Screen (paramètre DefaultColorDepth).

1.3. Xinerama

L'option Xinerama du ServerLayout n'est pas obligatoire. Son but est d'unifier les deux écrans en un grand écran. Elle permet de passer les fenêtres d'une application d'une fenêtre à l'autre, d'étendre une fenêtre sur deux écrans (ou plus). Cependant, il est tout à fait possible de ne pas vouloir cette option.

La désactivation de cette option rend les deux écrans indépendants, même si on ne se logge que sur un des deux pour accéder aux deux. Ceci peut permettre, quand on utilise plusieurs bureaux virtuels de changer de bureaux que sur un seul des deux

écrans.

2. Double écran sur un portable

Mon portable est un DELL LATITUDE D510 avec une carte vidéo intel i915GM dont le driver est le i810. Donc un "**man i810**" vous aidera beaucoup pour améliorer votre configuration.

2.1. Les sections Device

C'est sur ces sections que j'ai dû intervenir. Que ce soit Mandriva ou Ubuntu, personne n'était arrivé à me faire l'affichage sur les deux écrans alors qu'un essai sous l'OS concurent trop connu m'a montré que c'était possible.

Ce sont les deux lignes suivantes que j'ai du rajouter dans la partie device. La première indiquant de quelle sortie je parle (la 0 ou la 1), la seconde indiquant la nature des écrans. Cette seconde est aussi indispensable pour l'envoi simultané du même affichage sur l'écran du portable et sa sortie VGA. C'est encore elle que l'on doit modifier pour avoir la sortie TV (PAL indisponible sous Linux à l'instant où j'écris cet article).

2.2. Les sections ServerLayout

Dans cet exemple, contrairement à l'exemple précédent, plusieurs sections ServerLayout sont présentes. Ceci est assez rare et parfois bien utile.

2.2.1. Pourquoi différents ServerLayout

Avec un portable, on peut être dans plusieurs environnements suivant où on est. Il faut donc les prévoir, c'est pour celà que j'ai mis plusieurs ServerLayout. En effet, chacun défini un environnement de travail différent.

Le premier définit un affichage différent sur l'écran du portable et sur un écran extérieur, le mode double écran habituel. Ce mode, bien qu'utilisable avec un vidéo projecteur, n'est pas très pratique dans un tel environnement de travail.

Comme on n'a pas toujours d'écran extérieur disponible, le second défini un affichage uniquement sur l'écran du portable et comme ça ne demande rien de plus, j'impose en même temps une duplication sur la sortie extérieure afin de penser à un vidéo projecteur possible.

Le troisième active la sortie TV du portable afin de sortir sur la TV en mode SVideo.

2.2.2. Choix d'un ServerLayout

Par défaut, c'est le premier ServerLayout qui est choisi. Mais, dans la section **ServerFlags**, on peut en choisir un particulier avec **Option "DefaultServerLayout" "layout-id"**. Mais ces options étant en dur dans le fichier xorg.conf, ceci ne permet pas de changer facilement de configuration d'affichage !

Sous Mandriva, mon portable ne lance pas X au boot, je le fais moi-même à la main. Comme ça, je peux donc choisir la configuration graphique. Voici les commandes qui me servent à lancer le serveur X avec mon utilisateur lambda une fois logué en ligne de commande :

Les interfaces graphiques disponibles sous Mandriva sont visibles dans /etc/X11/wmsession.d/ .

Par défaut, je lance **Ice**, quand je souhaite un affichage unique, je choisi **seul**. Je peux même choisir **loin** afin de ne pas lancer le startup de IceWM qui râle quand je n'ai pas de connexion internet avec firefox et surtout thunderbird. À vous d'adapter ces alias si iceWM n'est pas votre interface graphique préférée !

Je suis sans voie pour les autres distributions. Je peux juste dire que pour Debian, ça ne marche pas et que je serais heureux que l'on me donne la solution !

2.3. Beryl 3D

Il est entièrement possible de faire marcher les bureaux 3D avec Beryl avec une carte graphique Intel, mais sachez que ceci est incompatible avec l'utilisation simultanée de 2 écrans configurée par xorg.conf .

Pour y arriver, il faut un peu configurer xorg.conf différement et rajouter des paquetages. Commençons par les paquetages de Mandriva :

Ces rpm, vous les trouverez peut-être sur les sources backport (selon un développeur : backport est censé être un dépôt où on met tout et n'importe quoi - surtout quand ça marche pas -) accessibles à partir de http://easyurpmi.zarb.org/?language=fr ou sur http://mandriva.beryl-project.org/2007/ (plus sur) avec le code suivant :
i586 dans mon cas ou x86_64 dans d'autres.

Dans la section "Module", vérifiez - ou ajoutez - les modules dri, dbe et glx.

```
Section "Module"
            # Load "GLcore"
            Load "bitmap"
            Load "ddc"
            Load "dbe"
            Load "dri"
            Load "extmod"
            Load "freetype"
            Load "glx"
            Load "int10"
            Load "type1"
            Load "vbe"
EndSection
```

Ajoutez ensuite, dans la section "Device", l'option "XAANoOffscreenPixmaps"

```
Section "Device"
            Identifier "deviceBeryl"
            Driver "i810"
            Option "XAANoOffscreenPixmaps"
            BusID "PCI:0:2:0"
EndSection
```

Rajoutez la section "ServerLayout" suivante avec l'option "AIGLX" activée :

```
Section "ServerLayout"
    Option "AIGLX" "true"
    Identifier "beryl"
    InputDevice "Keyboard1" "CoreKeyboard"
    InputDevice "Mouse1" "CorePointer"
    InputDevice "SynapticsMouse1" "AlwaysCore"
    Screen "deviceBeryl"
EndSection
```

On vérifie ensuite la section "DRI" qui devrait se présenter comme telle :

```
Section "DRI"
            Mode 0666
```

```
EndSection
```

Enfin, vérifiez la section "Extensions" :

```
Section "Extensions"
            Option "Composite" "Enable"
EndSection
```

Ensuite sous KDE ou Gnome, lorsque vous êtes sous le server Layout Beryl, lancez la commande

```
beryl-manager
```

3. Les essais

Il est rare que tout soit bon du premier coup !

3.1. xorg.conf

Je conseille fortement avec toute modification manuelle du fichier /etc/X/xorg.conf de le copier dans un répertoire dédié en rajoutant à son nom un suffixe pour mieux l'identifier.

Le point important à savoir quand on lance le gestionnaire de connexion, c'est que si dans le répertoire de lancement, il trouve un fichier de configuration de xorg, alors ce sera celui-ci qui sera utilisé. Donc, si vous modifiez le fichier /etc/X/xorg.conf et qu'il ne se passe rien, regardez ce qui se trouve dans le répertoire à partir du quel vous lancer X !

3.2. /var/log/Xorg.0.log

Le parcours du fichier de log peut être très intéressant en cas d'essais de modifications manuelles du fichier de configuration de xorg. Un **grep EE /var/log/Xorg.0.log** vous donnera la liste des options défectueuses et un **grep WW /var/log/Xorg.0.log** vous informera des petits défauts. Ceci peut permettre plus facilement de trouver d'où vient l'origine de votre problème et peut indiquer que des paramètres ne sont pas valables avec le driver que vous utilisez bien que vous les ayez crus universels. Je donnerais l'exemple de **Option "TV" "PAL-N"** qui n'existe pas avec le driver i810.

4. Améliorations possibles

Si vous avez lu mon document, vous êtes sûrement tombés sur des questions que je me pose, je suis bien sur preneur de toute réponse. Vous avez aussi vu que j'ai testé très peu de distributions : Ubuntu(que je n'utilise plus), Mandriva (que j'utilise principalement en version power pack ou cooker) ou Debian. Si vous testez sur d'autres distributions, dites-le moi !

Je ne suis pas un expert en la matière, j'ai juste passé du temps à tester la configuration de xorg pour arriver à mes fins. Comme j'y suis arrivé, j'ai fait ce texte pour vous aider si vous voulez y arriver, pour informer que c'est faisable si vous n'y pensiez pas, pour dire que sous Linux aussi, c'est faisable. Donc, si vous pensez que je dois rajouter des informations qui vous semblent importantes, faites-les moi parvenir pour que tous puissent en bénéficier.

La dernière mise au point que j'aimerai faire, c'est l'affichage en 1366x768 pour une TV 16/9 avec i915 resolution. Mon temps libre étant assez rare, j'espère que je pourrais tout de même vite y arriver !

Retrouvez l'article de Bernard Siaux en ligne : Lien1

Vu sur les Blogs

Nouvelle version du noyau Linux 2.6.19

La sortie du noyau 2.6.19 arrive 2 mois après la version 2.6.18. Cette nouvelle mouture rassemble plus de 5000 patchs et correctifs (pour la rc1 et la dernière release candidate est noté rc6) ! Au dire de Linus Torvalds, il s'agit d'un noyau de très bonne qualité : "It's one of those rare 'perfect' kernels" et, continuant sur le ton de l'humour, précise que si un de vos développements ne compile pas avec c'est entièrement de votre faute Lien2

Dans les nouveautés nous pouvons voir :

- la première version du système de fichier **ext4** (en expérimental)
- **GFS2** un système de fichier pour cluster développé par en grand partie par RedHat
- **eCryptfs**, développé par IBM, qui permet d'encrypter vos donnée (il ne s'agit pas d'un système de fichier crypter mais d'une surcouche au fs utilisé)
- gestion du **branchement à chaud** de CPU et de barrettes mémoire pour architecture x86-64
- un gestion amélioré pour les disques **PATA/SATA**
- une plétore de nouveau drivers pour **V4L** (Video for linux) carte tuner TV, acquisition vidéo etc..
- et plein d'autres choses...

Je vous invite à aller voir le résumé sur kernelnewbie : Lien3

Narmataru

Retrouvez le billet de Narmarataru : Lien4

Présentation de Java SE 6

Alors que **Java 5.0** s'annoncait comme une révolution, en apportant un grand nombre de modification dans le langage, le nouvel opus de **Java** se présente plus serein et mature. Pas de révolution pour **Java SE 6**, mais de vraies évolutions afin de préparer le futur du langage et son ouverture aux autres langages.

1. Avant propos...

Je tiens à remercier l'équipe de rédacteurs Java de developpez.com pour leurs conseils avisés...

2. Java Standard Edition 6

La prochaine version de **Java** ne devrait plus tarder à être disponible en version finale. Il est donc intéressant de prendre le temps de découvrir ce que nous apportera ce nouvel opus. Outre un nouveau changement de nom et de numérotation, ce qui devient presque une tradition puisqu'on est passé de **J2SE 1.4.2** à **J2SE 5.0** et que l'on a désormais droit à **Java SE 6** (on se demande ce qu'ils nous préparent pour **Java 7** !), cette nouvelle version a essayé d'apporter des solutions dans six grands domaines :

- **Compatibilité ascendante.**
  Bien que cela puisse paraitre évident pour beaucoup de monde, l'accent est mis sur la compatibilité ascendante. Bien que la plateforme soit mature, elle continue d'évoluer et continuera d'évoluer sans cesse, mais cela ne remet nullement en cause la compatibilité des programmes. N'importe quel programme qui fonctionnait sur une version précédente de la plateforme Java devrait fonctionner de la même manière avec **Java SE 6**.
- **Simplification du développement.**
  Java 5.0 a apporté des améliorations significatives avec l'introduction de nouveaux types et syntaxes, comme les types paramétrés (generics), les annotations, les énumérations ou encore les boucles for-étendus. Il reste toutefois encore beaucoup de travail pour simplifier la vie du développeur, en particulier en ce qui concerne l'accès aux bases de données, le support des langages de script, des technologies fondamentales telles que la compilation ou le traitement des annotations, ou encore un meilleur support des EDI et autres outils de développement.
- **Diagnostic, surveillance et gestion.**
  La plateforme Java est utilisée en production pour des applications critiques. **Java 5.0** avait apporté de nombreux progrès avec l'introduction de **JMX** (Java Management Extensions) et **JVMTI** (Java Virtual Machine Tools Interface) mais d'autres améliorations sont cependant nécessaires dans ce domaine.
- **Retour du "Desktop".**
  Les applications riches sont revenues sur le devant de la scène et les développeurs commencent à atteindre les limites des clients légers basés sur les navigateurs. **Java** est une alternative possible, mais elle souffre toujours de problèmes d'intégrations au sein du système de l'utilisateur final et certains composants souffrent de quelques gros défauts qui devront être comblés.
- **XML et Web Services.**
  Java 5.0 devait originellement proposer un ensemble d'outils facilitant la création et l'utilisation des Web Services. Malheureusement cela a pris plus de temps que prévu et cela n'a pas pu être intégré dans la spécification, et pendant ce temps XML et les Web Services ont pris de plus en plus d'importance pour beaucoup de membres de la communauté.
- **Transparence dans l'évolution.**
  La communauté a exprimé son désir de transparence dans l'évolution de la plateforme Java. Beaucoup voulaient pouvoir passer en revue et tester ces nouvelles spécifications alors même qu'elles étaient toujours en progrès, et devenir plus impliqué en contribuant à des améliorations ou des corrections de bugs. Le processus de développement de cette nouvelle version a donc été un des plus ouverts via son site communautaire : Lien5

3. Quoi de neuf docteur ?

Afin de déterminer les principales nouveautés de cette nouvelle mouture, il faut se tourner vers les JSR associées à cette version. Les JSRs (Java Specification Requests) représentent le processus normalisé via lequel sont définies les différentes spécifications des plateformes Java (Standard, Entreprise et Mobile).

Pour chaque JSR, un groupe de travail est créé. Il est composé de personnes provenant des différents acteurs du monde Java (comme la **fondation Apache**, **IBM**, **BEA Systems**, **Borland** et bien d'autres), qui coopèrent afin de définir ces spécifications. Ces dernières sont soumises à un vote afin d'être acceptées et intégrées dans le langage.

Avant de commencer, je tiens à préciser que ce document ne se veut pas exhaustif du tout, mais simplement une présentation brèves des nouveautés que cela nous apportera. En effet, étant donné la taille (et la complexité) des différentes spécifications, l'étude approfondie de chacune de ces JSRs pourrait faire l'objet de plusieurs articles. Cet article se contente donc de les citer et de les décrire brièvement. Pour plus de détails je vous conseille vivement de consulter les spécifications disponibles sur la page de chaque JSR.

JSR 223 : Scripting for the Java Platform

La **JSR 223** définit un framework permettant d'utiliser n'importe quel langage de script au sein d'une application Java. Elle permet ainsi de connaitre la liste des interpréteurs disponible sur la machine virtuelle, d'invoquer des scripts depuis une application Java (et vice-versa), de rendre les objets Java visibles et utilisables par ces scripts.

Par exemple, le code suivant permet grâce à l'interpréteur **JavaScript**, d'exécuter un script et de récupérer le résultat :

```
int intValue = 5;

ScriptEngineManager manager =
new ScriptEngineManager();
// On récupère l'interpréteur de script JavaScript²
ScriptEngine engine =
manager.getEngineByName("JavaScript");

// On crée une association pour la variable Java
'intValue'
engine.getBindings(ScriptContext.ENGINE_SCOPE).put(
"intValue", intValue);
// Evaluation d'un script :
Object result = engine.eval("(15 + intValue) / 2
");
// Affichage du résultat :
System.out.println("Résultat : " + result);
```

Ce qui nous donnera comme résultat :

```
Résultat : 10.0
```

Toutefois, il faut savoir que les spécifications n'imposent pas le support d'un langage de script en particulier. Ainsi chaque machine virtuelle peut proposer le support du/des langage(s) de son choix, et il faudra utiliser l'API pour vérifier la présence de l'interpréteur de script. Malgré cela, la JVM de Sun proposera d'office le support de **JavaScript** via l'interpréteur **rhino** de la **fondation Mozilla**, il y a donc de fortes chances que ce dernier se retrouve sur la plupart des JVM.

A noter également l'existence du projet "scripting" qui se chargera d'implémenter et de référencer les interpréteurs de script compatibles avec cette JSR. Elle comporte déjà plus de 20 langages de script dont **Python**, **Ruby**, **BeanShell** ou encore **Groovy** : Lien6

Cette JSR, représentée par le package **javax.script**, permet donc théoriquement l'utilisation de n'importe quel langage de script au sein de vos applications Java, et offre ainsi un large panel de nouvelles perspectives.

Pour plus de détails sur cette JSR, vous pouvez télécharger les spécifications sur la page de la JSR : Lien7

JSR 199 : Java Compiler API

Cette API définit un service permettant aux programmes Java d'invoquer un compilateur Java directement dans une application Java, et de récupérer toutes les informations retournées par le compilateur (erreurs, warnings, messages, etc.).

Cette JSR ne devrait pas vraiment concerner directement le développeur "de base", mais cible principalement les éditeurs d'outils Java :

- Les implémentations de moteur **JSP** pourront l'utiliser afin de compiler ces dernières sans invoquer de programme externe. Les premiers tests avec **GlassFish** (l'implémentation de référence de **Java EE 5**) ont donné des temps de compilations trois fois plus rapides (toute la compilation s'effectue en mémoire sans opération d'entrée/sortie).
- Les IDE pourront utiliser le compilateur directement, en utilisant une API standardisée (ce qui n'était pas le cas jusqu'à maintenant).
- D'autres outils ou API auront la possibilité de générer des fichiers ***.class** via la génération et la compilation de code source à la volée.

Son API utilise le package **javax.tools** et vous trouverez ses spécifications sur la page de la JSR : Lien8

JSR 269 : Pluggable Annotation-Processing API

Cette API permet de traiter les annotations lors de la compilation des codes sources, en interaction avec le compilateur. Il ne s'agit pas d'une nouveauté en soit, puisqu'un tel mécanisme était présent dans **Java 5.0** avec **APT** (Annotation Processing Tool). Toutefois, son API et les possibilités offertes ont été étendues et standardisées.

Cette API autorisera l'écriture de "librairies intelligentes" qui communiqueront avec le compilateur afin de signaler des erreurs d'utilisation ou encore de générer dynamiquement des classes et/ou des fichiers de configuration.

Elle utilise les packages **javax.annotation.processing** (pour le traitement des annotations à l'exécution) et **javax.lang.model.*** (qui représente le code lors de sa compilation).

Pour plus de détail : Lien9

JSR 250 : Common Annotations

Cette JSR vient définir cinq nouvelles annotations standard qui pourront également être utilisées par la plateforme entreprise (**Java EE**).

- **@Generated** servira à marquer les codes (ou les portions de codes) qui ont été générés par un outil, afin de les différencier de ceux qui ont été écrit par un développeur.
- **@PostConstruct** et **@PreDestroy** sont liée à la persistance des données, et permettent respectivement de définir une méthode qui sera appelée soit après la construction de l'objet (et éventuellement après l'injection de données), soit juste avant sa destruction.
- **@Resource** permet d'associer des ressources à des composants (classes, attributs ou méthodes), qui seront automatiquement injectées par les outils de déploiements. L'annotation **@Resources** quant à elle permettra simplement d'associer plusieurs ressources à un même composant.

Encore une fois, cette JSR n'est pas directement destinée aux développeurs mais aux outils divers, qui devront les utiliser pour simplifier la vie du développeur.

Le détail des spécifications est librement téléchargeable : Lien10

JSR 202 : Class File Specification Update

Le format des fichiers ***.class** a été mis à jour afin de supporter la "split verification", une architecture déjà présente dans la plateforme mobile de Java (Java ME), et qui apporte des avantages inhérents aux performances, aussi bien en terme d'espace utilisé (90%) que du temps de chargement de ces dernières (approximativement 2 fois plus rapide). Ce schéma est également plus simple et plus robuste, et facilitera les évolutions futures de la plateforme.

Bien entendu, tout cela est transparent pour le développeur, et la compatibilité ascendante reste d'actualité.

Toutes les informations sur ce nouveau format sont librement téléchargeables : Lien11

JSR 221 : JDBC 4.0 API Specification

Les spécifications de l'**API JDBC 4.0** viennent apporter un vent de fraicheur sur l'API d'accès aux bases de données de Java, en utilisant les facilités du langage apportées par Java 5.0 (Generics et Annotations en tête). Elle met l'accent sur les éléments suivant :

- Chargement automatique des drivers JDBC en utilisant le mécanisme de fournisseur de service.
- Utilisation d'annotations permettant de manipuler les tables facilement via un mapping object/relationnel très simple. Il est désormais possible d'exécuter des requêtes XML en écrivant un minimum de code Java.
- Support du type **SQL ROWID**, qui donne la possibilité de stocker l'index d'une ligne d'une table de la base de données.
- Amélioration du support des **BLOB** (Binary Large Object) et **CLOB** (Character Large Object).
- Support du type **SQL XML** défini dans le standard **SQL 2003**.
- Le gestionnaire de connexion a été amélioré et permet de détecter les connections devenues invalides.
- La hiérarchie des exceptions a été revue, et la classe **SQLException** possède désormais deux classes filles de base dont hériteront toutes les exceptions liées à JDBC. Ainsi **SQLTransientException** représentera les exceptions "passagères" (comme les problèmes de connexion, les timeouts). A l'inverse, les exceptions qui hériteront de **SQLNonTransientException** indiqueront un problème plus grave lié à une erreur irrémédiable qui devra être corrigée (argument invalide, erreur de syntaxe, etc.). De plus, toutes les **SQLException** sont désormais itérables et peuvent donc être utilisées dans une boucle for afin de dépiler la chaîne d'exception :

```
catch (SQLException e) {
    for (Throwable cause : e) {
        cause.printStackTrace();
    }
}
```

Les spécifications de **JDBC 4.0** sont bien sûr également disponibles : Lien12

Les JSRs concernant XML et Web Services

Comme prévu, **Java SE 6** sera fortement axé vers XML et les Web Services, avec ni plus ni moins que 5 JSRs sur le sujet, brièvement présentées ici :

- **JSR 105 : XML Digital-Signature API**

  Cette API, représentée par le package **javax.xml.crypto**, est une implémentation de la norme **XML Digital-Signature** du **W3C** (Lien13) et est une clef importante de la sécurité des Web Services. De plus elle est requise par d'autres JSR de cette section. Plus d'info : Lien14

- **JSR 173 : Streaming API for XML (StAX)**

  StAX est une API permettant d'exploiter les documents XML tout comme **DOM** ou **SAX** dont elle reprend les principaux avantages. **StAX** lit le document de manière linéaire afin de ne pas encombrer la mémoire (comme **SAX**), par contre la lecture du document n'est pas évènementielle mais c'est l'application qui détermine les éléments à lire (à l'instar de **DOM**). Bref **StAX** prend le meilleur des deux...

  Plus d'info : Lien15

- **JSR 181 : Web-Services Metadata for the Java Platform**
  Cette JSR définit un ensemble d'annotations permettant de décrire facilement des Web-Services au niveau du code source. Ces annotations détermineront comment les serveurs d'applications déploieront ces Web-Services. Plus d'info : Lien16

- **JSR 222 : Java Architecture for XML Binding (JAXB) 2.0**

  Il s'agit d'une révision majeure des spécifications de **JAXB 1.1**, l'API permettant de générer des classes Java à partir de schéma XML et inversement. Cette version met l'accent (entre autres) sur le support de la norme des **XML Schema**.

  Plus d'info : Lien17

- **JSR 224 : Java API for XML-Based Web Services (JAX-WS) 2.0**

  Cette spécification étend **JAX-RPC 1.0** afin de simplifier le développement, de mieux s'intégrer avec **JAXB 2.0** et d'apporter le support des derniers standards (**SOAP 1.2**, **WSDL 2.0** et **WS-I Basic Profile 1.1**). Plus d'info : Lien18

4. Et les APIs existantes ?

En plus de ces nouvelles spécifications, les APIs existantes ont également eut leurs lots de nouveautés et de modifications.

AWT et l'intégration au système

La mauvaise intégration des applications Java dans le système d'exploitation est un des points noirs des applications Java. Cette nouvelle version comble un peu ce déficit, notamment en ce qui concerne l'intégration au système :

La classe **java.awt.SystemTray** permet d'ajouter une icône dans la zone de notification du système (la plupart du temps il s'agit des icônes à coté de l'horloge) afin d'interagir avec l'utilisateur (clic, menu déroulant, affichage d'infobulle évoluée).

La classe **java.awt.Desktop** permet d'intégrer l'application au bureau du système d'exploitation, et permet trois principales actions :

- Ouvrir le navigateur par défaut sur une page précise (spécifiée par une **URI**).
- Ouvrir la fenêtre de saisie d'email du client de courrier électronique par défaut (éventuellement en y renseignant les différents champs via une **URI** de type **mailto:**).
- Utiliser l'association de fichier du système d'exploitation afin de lancer le programme associé pour effectuer des actions de base (ouvrir, éditer et imprimer).

Les boites de dialogues bénéficient d'un nouveau système de modalité plus élaboré. Ainsi on distingue désormais quatre types de modalité représentés par l'énumération **ModalityType** :

- **MODELESS** : La boite de dialogue n'est pas modale, elle ne bloquera donc aucune autre fenêtre.
- **APPLICATION_MODAL** : La boite de dialogue bloquera toutes les autres fenêtres et boites de dialogues de l'application (sauf ses éventuelles fenêtres filles). Il s'agit du comportement actuel des boites de dialogues modales.
- **DOCUMENT_MODAL** : La boite de dialogue bloquera toutes les fenêtres de la même hiérarchie, c'est à dire toutes ses fenêtres parentes, mais ne bloquera pas les autres fenêtres de l'application.
- **TOOLKIT_MODAL** : La boite de dialogue bloquera toutes les fenêtres du même toolkit (sauf ses éventuelles fenêtres filles). Ainsi par exemple toutes les applets d'un navigateur sont exécutées dans le même toolkit.

Il est également possible de définir, indépendamment pour chaque fenêtre, des exceptions à ces règles (représentées par

l'énumération **ModalExclusionType**) :

- **NO_EXCLUDE** : La fenêtre ne bénéficiera d'aucune exclusion et sera bloquée selon les règles en vigueur.
- **APPLICATION_EXCLUDE** : La fenêtre ne sera pas bloquée par les boites de dialogues de type **APPLICATION_MODAL**.
- **TOOLKIT_EXCLUDE** : La fenêtre ne sera pas bloquée par les boites de dialogues de type **TOOLKIT_MODAL**.

A première vue ces fonctions n'ont rien de révolutionnaire et pourraient même surprendre les développeurs non-java puisqu'il s'agit d'éléments qui peuvent sembler basiques. Il s'agit pourtant de notions qui ne sont pas forcément présentes sur tout les systèmes d'exploitations, et qui n'étaient donc pas intégrées dans l'API standard afin de ne pas casser la sacro-sainte loi *"Write once, run anywhere"* (*"Ecrire une fois, exécuter partout"*).

Afin de ne pas trop mettre à mal cette règle, et étant donné que ces fonctionnalités peuvent carrément être absentes du système d'exploitation, il est à la charge du développeur de vérifier dynamiquement le support de ces éléments via des méthodes adéquates, ainsi :

- Avant d'ajouter une icône dans la zone de notification, il faut vérifier que cela est bien supporté par le système avec la méthode **SystemTray.isSuported()**.
- Avant d'utiliser la classe **Desktop**, il faut vérifier si elle est supportée avec la méthode **Desktop.isDesktopSupported()**, puis pour chacune des actions possibles avec la méthode **Desktop.isSupported(Action)**.
- Avant d'utiliser un type de modalité ou d'exclusion, il faut vérifier que celui-ci soit supporté par le système avec les méthodes **Toolkit.isModalityTypeSupported(ModalityType)** et **Toolkit.isModalExclusionTypeSupported(ModalExclusionType)**.

Si la fonctionnalité est utilisée alors qu'elle n'est pas supportée, le comportement varie selon les cas. Ainsi pour les classes **Desktop** et **SystemTray**, une exception sera remontée, alors que lors de l'utilisation des types de modalité ou d'exclusions une valeur par défaut sera utilisée à la place (respectivement **MODELESS** et **NO_EXCLUDE**). Ainsi, il est à la charge du programmeur de proposer une méthode alternative si besoin.

Il est intéressant de noter que le même principe a été appliqué pour la méthode **setAlwaysOnTop()** introduite dans **Java 5.0** avec l'apparition des méthodes **Window.isAlwaysOnTopSupported()** et **Toolkit.isAlwaysOnTopSupported()**.

Note : Toutes ces fonctionnalités concernent le package **java.awt**, mais elles s'appliquent également aux applications **Swing** (n'oublions pas que **Swing** "hérite" d'**AWT**).

Swing - Les principales nouveautés

Swing a également subit pas mal d'améliorations, dont voici les principales :

- Tout d'abord les LookAndFeels natif **Windows** et **GTK** ont subit de nettes améliorations. Ils ne se contentent plus de recopier l'apparence du système, mais utilisent leurs API systèmes respectifs afin de récupérer les divers éléments de l'interface. Ils s'intègrent donc parfaitement dans l'environnement de l'utilisateur quel que soit son thème. Bien sûr cela fonctionne parfaitement sous **Windows Vista**.
- La classe **java.awt.SplashScreen** permet de gérer un "splash screen" natif au démarrage de l'application en spécifiant une image (via le manifest du jar ou une option de la ligne de commande). Ce dernier sera automatiquement détruit lorsque la première fenêtre de l'application sera ouverte.
- L'API destinée aux **LayoutManager** et aux outils d'interface graphique permet désormais de connaitre la ligne de base et les espacements préférés des composants, afin de mieux les disposer. De plus la famille des **LayoutManager** standard est agrandie par l'arrivée de **javax.swing.GroupLayout**, le manager utilisé par **Matisse**, le GUI-Builder de **NetBeans**, qui permet de grandement simplifier la création d'interfaces graphiques **Swing**.
- Tous les composants textes de **Swing** (héritant de **JTextComponent**) bénéficient désormais d'un support évolué de l'impression qui leur manquait cruellement.
- Il est dorénavant possible d'utiliser n'importe quel composant **Swing** en guise d'onglets sur les **JTabbedPanes** grâce à la méthode **setTabComponentAt()**.
- Les données des **JTables** pourront être triées ou filtrées simplement via **javax.swing.RowSorter** et **javax.swing.RowFilter**.
- Le **drag and drop** (glisser-déposer) a été grandement amélioré et simplifié afin de permettre d'utiliser plusieurs modes de fonctionnement via l'énumération **javax.swing.DropMode**.
- La classe **SwingWorker** permettant une gestion facile des traitements sans bloquer l'interface graphique est enfin intégrée en standard (cette classe faisait partie des tous premiers tutoriels officiels sur **Swing**).
- A ce propos, l'affichage via le double-buffer de **Swing** comblera désormais en partie le bug du rectangle gris ("gray rect") lorsque vous bloquez le thread d'affichage (**EDT**) : la fenêtre sera quand même complètement dessinée et n'affichera plus l'affreux rectangle gris (mais bien sûr votre interface sera quand même figée).
- La qualité de l'affichage des textes a été améliorée, notamment en utilisant la configuration système en ce qui concerne l'anti-aliasing.
- Le pipeline de rendu **OpenGL** (intégré avec **Java 5.0**) a été revu afin d'utiliser un mode **STR** (Single-Threaded Rendering).

L'API de Collection

Du coté de l'API de Collection, on note l'arrivée de cinq nouvelles interfaces :

- **Deque** : une queue à double sens, qui étend l'interface **Queue** de **Java 5.0** afin de supporter les insertions/suppressions à ses deux extrémités.
- **BlockingDeque** : une **Deque** avec des opérations bloquantes lors de l'ajout et suppression d'élément.
- **NavigableSet**, qui étend **SortedSet** afin de permettre une navigation en ordre croissant ou décroissant et des méthodes de recherches évoluées.
- **NavigableMap**, qui étend **SortedMap** afin de permettre une navigation en ordre croissant ou décroissant et des méthodes de recherches évoluées.
- Enfin, **ConcurrentNavigableMap** permet les mêmes possibilités tout en étant naturellement sécurisée en environnement multithread.

Cela s'accompagne également de nouvelles implémentations concrètes :

- **ArrayDeque** : Une implémentation de **Deque** basé sur l'utilisation de tableau redimensionnable.
- **ConcurrentSkipListSet** et **ConcurrentSkipListMap**, implémentent respectivement **NavigableSet** et **ConcurrentNavigableMap**.
- **LinkedBlockingDeque** : Une implémentation de **BlockingDeque** basé sur une liste chainée.

Sans oublier l'existence de deux implémentations de **Map.Entry** qui devrait faciliter l'écriture d'implémentation de **Map** personnalisée :

- **AbstractMap.SimpleEntry** est une implémentation de base.
- **AbstractMap.SimpleImmutableEntry** est une implémentation immuable (dont on ne peut pas modifier le contenu après la création).

A noter également la présence des méthodes **Arrays.copyOf()** et **Arrays.copyOfRange()** permettant de redimensionner/tronquer/copier des tableaux de tous types plus simplement qu'avec la méthode **System.arraycopy()** quelque peu complexe...

Entrées/Sorties et Réseaux

- La classe **java.io.Console**, dont l'unique instance est accessible via la méthode **System.console()**, nous permet une gestion un peu plus fine de la console, comme la lecture de mot de passe sécurisé et l'accès aux objets **Reader**/**Writer** associés. On regrettera toutefois l'impossibilité de manipuler le curseur et/ou d'effacer certaine portion de texte.
- La classe **File** permet désormais d'obtenir plus d'informations concernant l'utilisation du système de stockage, avec les méthodes suivantes :
    - **getTotalSpace()** indique la taille totale de la partition (en bytes).
    - **getFreeSpace()** indique la taille disponible sur la partition (en bytes).
    - **getUsableSpace()** indique la taille utilisable sur la partition (en bytes). Ceci inclut la vérification des droits d'accès et d'éventuelles restrictions systèmes.
- De plus, il est désormais possible de modifier les attributs des fichiers (pour l'utilisateur courant ou pour tout le monde) via les méthodes **setWritable()**, **setReadable()** et **setExecutable()** de la classe **File**. En toute logique, la réussite de ces méthodes dépend du système d'exploitation.
- **java.net.IDN** apporte le support des noms de domaine internationalisé (**IDN**) au format Unicode et de leurs équivalent ASCII.
- L'interface **java.net.NetworkAddress** a été étendue afin de fournir plus d'informations sur le réseau (adresse de broadcast, masque réseau, adresse MAC, taille du MTU, état activé ou désactivé, etc.), notamment via l'interface **java.net.InterfaceAddress**.
- **Java 5.0** avait introduit la classe abstraite **java.net.CookieHandler** permettant de gérer les Cookies lors des connections HTTP, mais ne proposait aucune implémentation par défaut. C'est désormais chose faite avec **java.net.CookieManager** qui propose une implémentation qui devrait suffire à la plupart des applications.

Localisation et internationalisation

- *Pluggable locale* : La gestion de locale du JRE permet désormais l'installation de Locale non-supportée via un système de plugin.
- Les **ResourceBundles** ont bénéficié de quelques améliorations en proposant un dispositif permettant de contrôler la gestion et le comportement du cache, ou encore le support de n'importe quel type de formats en entrée (comme XML par exemple).
- La classe **java.text.Normalizer** permet la transformation de chaines de caractères Unicode dans une forme composée ou décomposée, selon les standards de l'Unicode. En utilisant la même forme d'encodage, on peut manipuler les chaines avec des caractères spéciaux ou des alphabets exotiques plus simplement. A titre d'exemple, le caractère **À** peut être représenté par la forme composée **\u00C1** (*A majuscule avec accent grave*) ou par la forme décomposée **\u0041\u0301** (*A majuscule combiné avec un accent grave*). En utilisant la même forme pour toutes les chaines on facilite la comparaison de ce type de formation.
- Ajout du support du calendrier impérial chinois et de nombreuses nouvelles locales (zh_SG, en_MT, en_PH, en_SG, el_CY, id_ID, ga_IE, ms_MY, mt_MT, pt_BR, pt_PT, et es_US).

JVMTI (JVM Tool Interface)

- Les outils d'analyses auront dorénavant la possibilité d'accéder au contenu de la mémoire heap via **JVMTI**.
- **Attach-on-demand** : **Java 5.0** avait apporté la possibilité de surveiller le comportement d'une JVM lorsqu'elle était lancée avec certains paramètres. Il sera désormais possible de surveiller n'importe quelle application Java quel que soit la manière dont elle a été démarrée (et donc de démarrer une surveillance sur une application sans la redémarrer).

Divers

- Calcul à virgule flottante : ajout de méthodes recommandées par l'**IEEE 754**. Ainsi les classes **Math** et **StrictMath** se sont vue rajoutées les méthodes **scalb()**, **getExponent()**, **nextAfter()**, **nextUp()**, et **copySign()** pour les types **float** et **double**.
- Le package **java.util.concurrent** introduit dans **Java 5.0** a reçu quelques correctifs mineurs, comme l'ajout de nouvelles unités de temps dans la classe **TimeUnit** : **MINUTES**, **HOURS** et **DAYS**.
- La nouvelle classe **java.util.ServiceLoader** permet de gérer un système de service tel qu'il est déjà utilisé par plusieurs classes de l'API standard, en recherchant toutes les implémentations concrètes d'une interface (ou d'une classe abstraite) déclarée dans un des fichiers des répertoires **META-INF/services** de chacun des éléments du **CLASSPATH**. Ceci est utilisé par exemple pour rechercher automatiquement les drivers **JDBC**.
- On retiendra également le support des images au format GIF en écriture (jusqu'à présent seul le support en lecture était disponible en standard) qui est devenu possible avec l'expiration des brevets le concernant (le format est dorénavant dans le domaine public).
- Enfin la définition du **CLASSPATH** supporte désormais le wildcard * qui représente toutes les archives **jar**/**zip** d'un répertoire. Ainsi si le **CLASSPATH** (via la ligne de commande, la variable d'environnement ou l'attribut du manifest du Jar) comporte par exemple **lib/*** toutes les archives du répertoire **lib** seront ajoutées au Classpath de l'application, et il sera donc inutile de les nommer une à

une.

Pour aller plus loin...

Si le sujet vous intéresse et que vous voulez approfondir les choses, je ne peux que vous conseillez d'aller jeter un coup d'oeil aux différentes JSRs ou sur le site officiel :

- **JavaTM SE 6 Release Notes - Features and Enhancements** : Lien19
- **JDK 6 Documentation** : Lien20
- **Java Platform API Specification** : Lien21

Et pourquoi pas de venir en parler avec nous sur le forum **Java** de **Developpez.com** : Lien22

Lire l'article de Frédéric Martini en ligne :Lien23

Vu sur les blogs

Retrouvez tout Javapolis 2006 grâce à vincent brabant !

C'est tout simplement le meilleur Javapolis que je n'ai jamais vécu (c'est ma 4ième session).

Nous avons fait la connaissance de Juggy, la mascotte des Java User Group. Juggy nous a présenté à un certain nombre de personnes. Et nous avons pu réalisé, avec sa participation, quelques interviews de personnes que nous n'aurions jamais pu interviewer autrement.

Ces interviews ont été réalisés en Anglais, mais nous aurons certainement l'occasion de vous les proposer avec des sous-titres.

Par contre, nous avons également pu faire deux interviews, dont celui de Marc Fleury, patron de JBoss, en français. Ce qui fut, pour moi, un vrai plaisir.

Mercredi soir, nous avons assisté au BOF "Meet the Java-champions", ce qui fut pour moi l'occasion d'en revoir certain que j'avais déjà croisé l'année passée (Forza Italia), mais aussi d'en croiser de nouveaux. Et pour Christophe, de faire la connaissance d'une bonne partie de l'équipe de java-champions.

Jeudi soir, nous avons assisté au JUG Leader BOF, en tant que représentant de java.developpez.com qui est un Virtual Java User Group.

D'ailleurs, si quelqu'un se sent d'attaque pour aider à ce que java.developpez.com puisse également devenir un vrai Java User Group, en organisant des rencontres et des évènements Java, qu'il se présente.

Nous ne pourrons jamais assez remercier Bruno Souza (leader du JUG au brésil, celui qui est derrière Juggy), pour nous avoir introduit auprès de toutes ces personnes, ainsi que Juggy qui a joué le rôle d'interviewer auprès de ces personnes.

On vous racontera les sessions plus tard.

Retrouvez les billets de vbrabant en ligne : Lien24

Upload de fichiers en PHP

Vous avez envie de permettre aux visiteurs de votre site d'uploader des fichiers sur votre serveur ? Mais vous ne savez pas comment faire ?
Alors ce tutoriel est fait pour vous. Vous verrez, en lisant ce tutoriel qu'uploader un fichier en php est faisable mais qu'en plus c'est très simple ! ;o)

1. Préface

Bonjour, à tous, dans cet article, vous allez apprendre à uploader un fichier grâce à un formulaire sur une page web ! L'upload est l'opération qui consiste à envoyer des informations locales (provenant de votre disque dur) sur le web. Par exemple, un tel script peut s'avérer utile pour permettre aux membres de votre site de créer leur avatar...

Le tutoriel se déroulera en 2 parties : le formulaire et l'upload. Pour simplifier, nous allons diviser le script en deux fichiers...

2. Le formulaire

Pour commencer, créez une page "upload.html". Cette page contiendra le formulaire.

2.1. Le formulaire

Lorsque vous souhaitez envoyer un fichier au serveur par formulaire, vous devez préciser l'enctype, c'est à dire le type d'encodage du fichier. L'enctype à utiliser est multipart/form-data. C'est très important car si vous ne le faites pas, vous ne pourrez pas uploader votre fichier !

Votre formulaire vide donnera donc :

Code pour ouvrir et fermer le formulaire

```
<form method="POST" action="upload.php"
     enctype="multipart/form-data">
     <!-- Le contenu du formulaire -->
</form>
```

2.2. Le champ fichier

Pour trouver le fichier, vous devez permettre à l'utilisateur de choisir un fichier sur son disque dur.
Heureusement, HTML le fait pour nous ! ;o)
Le champ de type "file" a donc cette mission.

Code pour insérer un champ fichier

```
<input type="file" name="avatar">
```

Comme pour tous les champs de formulaire, vous devez lui donner un nom.
Je vais mettre "avatar" pour l'exemple.

2.3. Limiter la taille du fichier

Vous aurez sûrement besoin de limiter la taille du fichier.

Il ne faudrait pas que quelqu'un s'amuse à uploader des images de plusieurs Mo...

Pour cela, vous devez utiliser un champ caché "hidden", lui donner le nom "MAX_FILE_SIZE" et lui donner en valeur, la taille maximum du fichier à uploader en octets.

Code pour limiter la taille du fichier

```
<input type="hidden" name="MAX_FILE_SIZE"
value="100000">
```

2.4. Le formulaire complet

Bon allez, je vous donne le formulaire complet :

Code final du xHTML du formulaire

```
<form method="POST" action="upload.php"
       enctype="multipart/form-data">
       <!-- On limite le fichier à 100Ko -->
       <input type="hidden" name="MAX_FILE_SIZE"
       value="100000">
       Fichier : <input type="file" name="avatar">
       <input type="submit" name="envoyer"
       value="Envoyer        le fichier">
</form>
```

Maintenant que vous avez le formulaire en entier, vous pouvez constater que les données seront envoyées à "upload.php".

Et bien maintenant, nous allons nous occuper d'upload.php !

3. L'upload

Bien ! Nous allons maintenant attaquer la partie la plus intéressante, à savoir l'upload.
Nous considérons que nous venons de la page upload.html.
Lorsque le formulaire a été validé, le fichier a été placé dans un dossier temporaire et sera supprimé si nous ne nous en occupons pas...

3.1. Récupérer le fichier envoyé

Heureusement pour nous, PHP a tout prévu !
Les informations sur les fichiers envoyés sont contenues dans le tableau global $_FILES.
$_FILES contient :

NOM	VARIABLE
Le nom	$_FILES['avatar']['name']
Le chemin du fichier temporaire	$_FILES['avatar']['tmp_name']
La taille (peu fiable, dépend du navigateur)	$_FILES['avatar']['size']
Le type MIME (peu fiable, dépend du navigateur)	$_FILES['avatar']['type']
Un code d'erreur si besoin	$_FILES['avatar']['error']

3.2. L'upload

Pour ce faire, nous allons avoir besoin de la fonction... move_uploaded_file()

Je vous laisse cliquer dessus pour voir la doc...

Cette fonction renvoie TRUE si elle a correctement fonctionné et FALSE si elle n'a pas correctement fonctionné.

Nous allons aussi avoir besoin d'un dossier, upload, auquel vous aurez généreusement au préalable accordé les droits d'écriture grâce à votre logiciel FTP.

Pour accorder les droits d'écriture, vous devez changer le CHMOD du dossier upload. Pour changer le CHMOD avec votre logiciel FTP, cliquez droit sur le dossier upload et cliquez sur "Attribut du fichier" ou "Options du CHMOD" (le nom varie selon les logiciels...). Le CHMOD que nous devons utiliser est le 755. Je vous fournirai un lien vers la FAQ expliquant le CHMOD à la fin de ce tutoriel... ;o)

Voici donc le code d'upload que je vais vous expliquer :

Code d'upload de base

```
<?php
if(isset($_FILES['avatar']))
{
     $dossier = 'upload/';
     $fichier = basename($_FILES['avatar']['name']);
     if(move_uploaded_file($_FILES['avatar']['tmp_name'
], $dossier . $fichier)) //Si la fonction renvoie TRUE,
c'est que ça a fonctionné...
     {
          echo 'Upload effectué avec succès !';
     }
     else //Sinon (la fonction renvoie FALSE).
     {
          echo 'Echec de l\'upload !';
     }
}
?>
```

Comme vous pouvez le constater, ce n'est pas bien compliqué... Je vais vous laisser cogiter sur ce code tout seuls...

3.3. La sécurité

STOP! Ne mettez surtout pas ce script en ligne ! ! !

Rassurez-vous, ce n'est pas fini, ce serait trop simple sinon... Il ne faut quand même pas exagérer !

3.3.1. Le type de fichier

Qui vous dit que l'utilisateur n'est pas en train d'uploader un fichier PHP par exemple pour tout supprimer ou récupérer le contenu de vos tables (mots de passe, adresses email...).
C'est pourquoi il faut vérifier le type du fichier que vous voulez uploader.
Voici la procédure :

Code pour vérifier le type du fichier uploadé

```
<?php
//On fait un tableau contenant les extensions
autorisées.
//Comme il s'agit d'un avatar pour l'exemple, on ne
prend que des extensions d'images.
$extensions = array('.png', '.gif', '.jpg', '.jpeg');
// récupère la partie de la chaine à partir du dernier
. pour connaître l'extension.
$extension = strrchr($_FILES['avatar']['name'], '.');
//Ensuite on teste
if(!in_array($extension, $extensions)) //Si l'extension
n'est pas dans le tableau
{
     $erreur = 'Vous devez uploader un fichier de type
png, gif, jpg, jpeg, txt ou doc...';
}
?>
```

Cette méthode est une première approche qui nous suffit pour le moment mais, en réalité, il faudrait vérifier le type MIME des fichiers uploadés.

3.3.2 La taille du fichier

Un utilisateur peu scrupuleux peut enregistrer le formulaire sur son disque et modifier la valeur du champ "MAX_FILE_SIZE" !
Pour remédier à cela, nous devons vérifier la taille du fichier avant de l'uploader...
Je vous rappelle que le tableau global $_FILES contient la taille du fichier mais qu'elle est peu fiable.
Heureusement, la fonction PHP filesize() retourne la taille d'un fichier en octets. Il suffit de lui donner l'adresse du fichier qui est contenue dans $_FILES['avatar']['tmp_name'] donc pas de panique...
Voici le code :

Code pour vérifier la taille du fichier

```
<?php
// taille maximum (en octets)
$taille_maxi = 100000;
//Taille du fichier
$taille = filesize($_FILES['avatar']['tmp_name']);
if($taille>$taille_maxi)
{
     $erreur = 'Le fichier est trop gros...';
}
?>
```

3.3.3 Le nom du fichier

Le nom du fichier peut lui aussi poser problème...

Nous devons donc nous occuper de lui car s'il contient des accents, caractères spéciaux, espaces, ça peut poser problème.

Nous allons donc "formater" le nom du fichier avant de l'uploader...

Il s'agit d'une expression régulière. Je ne vais pas vous expliquer les expressions régulières dans ce chapitre car il s'agit d'une chose

assez difficile au début et il faudrait y consacrer au moins un cours entier.

Code pour formater le nom du fichier

```
<?php
$fichier = strtr($fichier,
     'ÀÁÂÃÄÅÇÈÉÊËÌÍÎÏÒÓÔÕÖÙÚÛÜÝàáâãäåçèéêëìíîïðòóôõöùúûüýÿ',
     'AAAAAACEEEEIIIIOOOOOUUUUYaaaaaaceeeeiiiioooooouuuuyy');
//On remplace les lettres accentutées par les non accentuées dans $fichier.
//Et on récupère le résultat dans fichier

//En dessous, il y a l'expression régulière qui remplace tout ce qui n'est pas une lettre non accentuées ou un chiffre
//dans $fichier par un tiret "-" et qui place le résultat dans $fichier.
$fichier = preg_replace('/([^.a-z0-9]+)/i', '-', $fichier);
?>
```

Voilà, c'est tout pour la sécurité fondamentale. ;o)

3.4. Le code final

Voici en exclusivité, le code d'upload !

Code final de l'upload

```
<?php
$dossier = 'upload/';
$fichier = basename($_FILES['avatar']['name']);
$taille_maxi = 100000;
$taille = filesize($_FILES['avatar']['tmp_name']);
$extensions = array('.png', '.gif', '.jpg', '.jpeg');
$extension = strrchr($_FILES['avatar']['name'], '.');
//Début des vérifications de sécurité...
if(!in_array($extension, $extensions)) {
     $erreur = 'Vous devez uploader un fichier de type png, gif, jpg, jpeg, txt ou doc...';
}
if($taille>$taille_maxi){
     $erreur = 'Le fichier est trop gros...';
}
if(!isset($erreur)) {//S'il n'y a pas d'erreur, on upload
     //On formate le nom du fichier ici...
     $fichier = strtr($fichier,
          'ÀÁÂÃÄÅÇÈÉÊËÌÍÎÏÒÓÔÕÖÙÚÛÜÝàáâãäåçèéêëìíîïðòóôõöùúûüýÿ',
          'AAAAAACEEEEIIIIOOOOOUUUUYaaaaaaceeeeiiiioooooouuuuyy');
     $fichier = preg_replace('/([^.a-z0-9]+)/i', '-', $fichier);
     if(move_uploaded_file($_FILES['avatar']['tmp_name'], $dossier . $fichier)) { //Si la fonction renvoie TRUE, c'est que ça a fonctionné...
          echo 'Upload effectué avec succès !';
     }
     else {//Sinon (la fonction renvoie FALSE).
          echo 'Echec de l\'upload !';
     }
}
else{
     echo $erreur;
}
?>
```

4. Conclusion

Voilà ! ! ! C'est fini ! Vous pouvez maintenant uploader des fichiers !
J'espère que vous avez trouvé mon petit article intéressant !
Si vous trouvez des erreurs, signalez-les moi, c'est comme ça qu'on avance...
Comme promis, voici un lien vers la FAQ au sujet du CHMOD : Lien25
Maintenant que vous savez uploader des fichiers, je vous propose un tutoriel pour les gérer : Upload de fichiers en PHP Lien26
Pour ceux qui voudraient en savoir un peu plus sur les formulaires et PHP5 :
Lien27

Retrouvez l'article de Antoine Herault en ligne : Lien28

TV

Forum AFUP PHP 2006

Jeudi et vendredi 9 et 10 Novembre, s'est tenu le forum AFUP 2006. Comme les années précédentes, ce fut un événement rondement mené, avec beaucoup d'excellentes conférences et de rencontres.

Côté contenu, Rasmus Lerdorf et Zeev Suraski étaient présents. Coté francophone, il y avait des retours d'expérience impressionnants dans l'administration belge, dans les décodeurs Canal Plus, ou encore avec une application de mesure de la qualité logicielle PHPMeter

Retrouvez le Reportage Complet : Lien29

Bon visionnage !!!

Visionnez les vidéos : Lien30

Livres Flash

Flash 8 Professinal De Mathieu Lavant

Critique par Grégory Dumas

Destiné aux graphistes comme aux développeurs Flash", ainsi commence l'accroche écrite au verso du livre. A cette description, je préfère nettement, celle faite par l'auteur dans l'avant-propos que je me permet de retranscrire ici :

« Ce livre s'adresse d'abord à tous les apprentis "flasheurs" étudiants ou autodidactes qui souhaitent acquérir rapidement une bonne maîtrise du logiciel. Il s'adresse ensuite aux webdesigners et graphistes qui utilisent déjà Flash et qui désirent connaître les nouveautés de la version 8. Il s'adresse enfin à tous les utilisateurs, non développeurs, qui souhaitent dépasser le b.a-ba de l'interactivité et enrichir leur création avec ActionScript. »

Ce livre décrit de manière très complète les possibilités offertes par Flash et donne toutes les bases pour bien appréhender le logiciel. Dans ce sens, les débutants mais aussi tous les webdesigners qui ne souhaitent pas forcément s'investir dans le coté programmation de Flash avec le langage ActionScript, tous ces utilisateurs trouveront dans cet ouvrage une aide précieuse.

Par contre, je déconseille ce livre aux personnes plus expérimentées ou pour quiconque qui souhaiterait dépasser la simple utilisation du logiciel et s'ouvrir vers de nouvelles perstectives qu'offre la programmation ActionScript. En effet, dans ce livre, pour en faciliter la compréhension, l'auteur prend le parti d'utiliser un "mode de programmation" que je qualifierai de désuet et qui me donne l'étrange impression de lire un livre vieux de 2/3 ans déjà. Il a préfèré donner des techniques aisément assimilables par le lecteur pour qu'il puisse rapidement mettre en application ce qu'il a appris et réaliser des applications flash interactives; charge alors à ce dernier de s'informer, se documenter pour que ces techniques ne deviennent pas rapidement de mauvaises habitudes.

En conclusion utile pour tous ceux qui souhaitent apprendre à maîtriser l'IDE Flash, beaucoup moins pour ceux souhaitent apprendre à programmer en ActionScript.

Flash 8 pour les webdesigners De Guylaine Monnier

Critique par Arnaud Lemercier

Flash 8 pour les webdesigners : vous allez l'adorer ou le détester. Avant tout, ne pensez pas que ce livre s'adresse à des futurs artistes, c'est plutôt un livre qui détaille les généralités de Flash 8.

Si vous avez déjà lancé le logiciel d'Adobe, vous allez surement vous ennuyer pendant les premiers chapitres. Cependant, on ne peut pas reprocher à l'auteur d'avoir couvert les aspects fondamentaux de Flash jusque dans les moindres détails.

Je pense que les débutants y trouveront leur compte et ne seront plus seuls face à Flash 8.

Guylaine Monnier n'a pas pour ambition de vous apprendre l'ActionScript. Ne cherchez donc pas à apprendre la programmation grâce à ce livre, il ne vous livrera que le minimum requis pour le contrôle de scénarios, la conception de sites, l'utilisation de médias (sons, vidéos, images)...

Vous allez acquérir une méthode de travail portant sur des optimisations constantes de vos projets. Malgré l'expérience de l'auteur, le livre reste trop théorique et manque profondément de cas réutilisables. Il vous garantit un apprentissage sérieux, la maitrise de l'interface du logiciel jusqu'à la construction de site ou d'animation.

Je le conseille à tous ceux qui souhaitent débuter Flash doucement mais surement.

Retrouvez tous les livres de la section Flash : Lien31

Analyser du code .Net avec FxCop

Ecrire du code n'est pas chose difficile, écrire du code dans "les règles de l'art" l'est déjà plus ! Un outil gratuit, et spécialisé dans l'analyse du code .Net, vous offre la possibilité de vérifier qu'il n'y a pas d'anomalie, ou même d'erreur grave, dans vos assemblies, et que celles-ci sont conformes aux "bonnes pratiques" recommandées par Microsoft. Cet outil se nomme FxCop, il fonctionne aussi bien avec les assemblies .Net 1.1 que 2.0, je vais essayer au cours de cet article de vous montrer l'avantage que vous pouvez tirer d'une telle application.

1. Introduction

Programmer, c'est comme écrire, il y a ceux qui ont du style, ceux qui n'en ont pas, ceux qui font des fautes, et les autres. Même si tout langage de programmation, et toute technologie impose des contraintes, des syntaxes et des règles précises, il n'empêche que la même application écrite par dix développeurs différents ne sera jamais exactement la même. Chaque personne à une manière différente de tourner des algorithmes, de nommer des variables ou je ne sais encore quel détail qui fait le style d'un développeur. Cependant, même si un code ne peut pas être strictement prédéfinit, il doit quand même satisfaire un certain nombre d'exigences, en termes de design et de syntaxe. Toutes ces règles et normes, sont définies pour apporter le meilleur niveau de performance et de standardisation possible aux applications. Il faut prendre ces termes au sens large, en effet, cela concerne les performances brutes, mais aussi la sécurité, les conventions de nommage, le design... Toutes ces mesures qui font qu'un programme est écrit dans les règles de l'art ou non. Rares, voir même inexistants, sont les développeurs capables d'écrire ce code "parfait" au premier jet, sans avoir à y revenir.

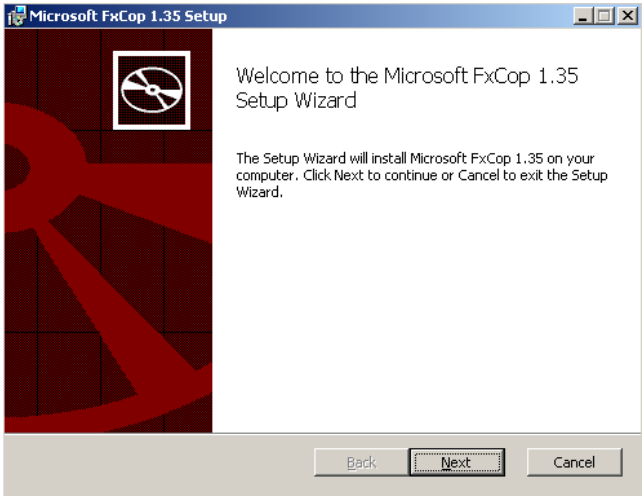
Ces règles et nomenclatures sont nombreuses, et parfois complexes. De plus, celles-ci ne sont pas gravées dans le marbre, et de ce fait peuvent évoluer, ou être différentes d'une société à une autre. Cet ensemble fait qu'il est complexe, coûteux en temps et peu motivant de parcourir ses lignes de code à la recherche des éléments à modifier pour se conformer aux différentes exigences. Et une fois de plus, le développeur a trouvé la parade, une application est capable de faire cela ! Elle se nomme FxCop. Bien entendu, ce n'est pas la seule application capable de réaliser une telle analyse, mais c'est l'une des plus connues, elle est gratuite, performante, et issue de la communauté des développeurs .Net. C'est donc pour toutes ces raisons que j'ai choisi de la présenter. J'avoue l'utiliser depuis quelques mois seulement, mais, elle me rend de fiers services en me procurant un code plus simple à maintenir, en évitant les grosses fautes, pouvant notamment avoir un impact sur les performances ou la sécurité d'une application. Avant de voir plus en détails FxCop, voyons tout d'abord où trouver ce logiciel, et comment l'installer.

2. Installer

FxCop, à l'heure où j'écris ces lignes est en version 1.35, et nécessite le Framework .Net 2.0. Cette version est capable d'analyser des assemblies issues de toutes les versions du Framework inférieures ou égales à la 2.0. Mais gageons que l'équipe de développement de FxCop saura réagir et adapter cet outil à la sortie prochaine du Framework 3.0, qui se fera parallèlement à la sortie de Windows Vista. Cet outil, gratuit je le rappelle, est disponible sur le site de GotDotNet. Par rapport à la version précédente (1.32), la 1.35 corrige plus de 150 bugs, rectifie certaines règles pouvant donner des résultats erronés, et ajoute cinq nouvelles règles

Cet outil est disponible en téléchargement (environ 3.6 Mo) à cette adresse : Lien32

Pour l'installation même du programme, vous allez voir, il n'y a rien de compliqué, c'est juste du "next next ok"... Bref, à la portée de tout le monde. Le seul prérequis est le Framework .Net 2.0, en effet cette dernière version en a besoin pour s'exécuter. Vous admettrez que pour une application dont le but est d'analyser des assemblies .Net cela n'a rien de révoltant. Voici quelques captures d'écrans de l'installation :



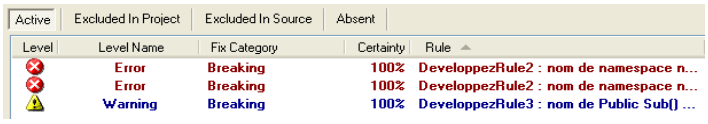
Remarque : petite précision, FxCop fonctionnera quelque soit l'environnement de développement que vous utilisez ! Cela va donc de Notepad à Visual Studio 2005 Team Suite, en passant par SharpDevelop. En effet, FxCop est totalement indépendant de ces logiciels, et fonctionne de manière autonome.

3. Présentation

Tout d'abord je vais vous présenter FxCop et ses fonctionnalités dans les grandes lignes. Histoire de comprendre comment il fonctionne, ce que l'on peut en attendre ou non. Premièrement, vous devez savoir que cet outil travaille sur des assemblies "compilées" et non sur le code source, c'est-à-dire que son domaine de prédilection c'est le MSIL et non le VB.Net, C#, ou encore Delphi.Net... Ce détail a son importance, travaillant sur des

assemblies et non sur du code source, FxCop est totalement indépendant vis-à-vis du langage de que vous avez employé pour votre programme, du moment bien évidemment qu'il s'agisse de .Net ! Mais est-ce bien utile de le préciser ?

Son principe de fonctionnement est le suivant : FxCop procède à une analyse méticuleuse du code MSIL d'une assembly, et vérifie que l'assembly est en conformité avec les règles qui lui sont associées. En effet, chaque point vérifié correspond en fait à une règle, qui, elle-même est issue des bonnes pratiques édictées par Microsoft. Bonnes pratiques, qui sont d'ailleurs publiquement connues, puisqu'en libre téléchargement sur le site MSDN. FxCop n'invente donc rien, mais compile et concentre en une seule application toutes ces bonnes pratiques. Quand on sait que la version 1.35 en contient plus de 200 on réalise alors la productivité apportée par un tel outil. Bien entendu, il est possible de choisir parmi toutes ces règles lesquelles doivent être vérifiées ou non, nous verrons même plus loin qu'il est possible de créer ses propres règles entièrement personnalisées. Une fois l'analyse complétée, dont la durée sera évidemment plus ou moins grande en fonction du nombre, de la taille, et de la complexité des assemblies chargées, un rapport est généré. Un premier tableau contient une liste comportant chaque anomalie détectée en précisant notamment le type, le niveau, le pourcentage de probabilité réelle du problème. D'autres informations sont également fournies, je vous laisse les découvrir. Le plus intéressant, concerne la partie basse de l'interface qui contient le détail de l'anomalie sélectionnée. Ici ce sont des informations très détaillées qui sont données, voici d'ailleurs en exemple une capture d'écran.



Liste des anomalies détectées

Détails d'une anomalie

Comme vous pouvez le voir ci-dessus, FxCop propose une solution permettant de corriger le problème, et aussi un lien pointant vers une page décrivant plus précisément l'anomalie détectée, j'ai constaté qu'il arrive cependant que la page en question ne soit pas encore disponible, mais cela ne se produit pas dans la majorité des cas. La solution proposée prend en compte la spécificité de votre code. Des informations complémentaires sont également données, il est donc généralement facile de cerner l'anomalie qui a été repérée.

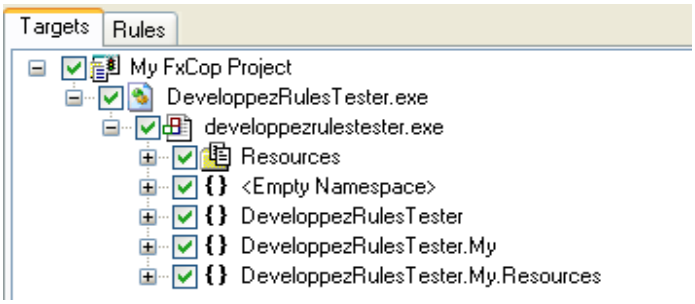
Les anomalies sont classées dans quatre catégories différentes :

- Warning
- Critical Warning
- Error
- Critical

Remarque : il faut préciser que, s'il est possible d'analyser plusieurs assemblies en même temps, celles-ci doivent utiliser la même version du Framework. Par exemple, analyser une assembly .Net 1.1 et une 2.0 en même temps ne sera pas possible.

4. Analyser ses assemblies

Après avoir étudié les principes de fonctionnement de FxCop, voyons maintenant comment procéder à l'analyse concrète d'assemblies. La première étape consiste à créer un nouveau projet d'analyse, pour cela il faut aller dans le menu "File" et choisir "New project". Ensuite, il faut sélectionner la, ou les assemblies à analyser, menu " Project ", "Add targets". FxCop charge alors le code à analyser. Voici une capture d'écran montrant à quoi doit ressembler le logiciel à cet instant.



Maintenant, nous allons choisir les règles et bonnes pratiques que nous souhaitons vérifier avec FxCop. Cela se passe dans l'onglet "Rules", dans la partie gauche de l'interface. Les règles sont divisées en neuf catégories qui sont les suivantes :

- **Design rules** : cette catégorie regroupe, comme son nom l'indique, les règles de design .Net. C'est-à-dire les manières de concevoir et de réaliser les différents objets, classes, membres... Cela reprend le contenu du ".NET Framework Design Guidelines" publié par Microsoft.
- **Globalization Rules** : ce sont des règles avec l'application de la globalisation du code et des applications. En relation notamment avec la culture, la langue et les paramètres propres à une région.
- **Interoperability Rules** : ce sont les règles qui vérifient les éléments du code relatif aux problèmes d'interaction avec les clients COM.
- **Mobility Rules** : ces règles vont vérifier qu'il n'y a pas de risque d'activité excessive de l'application, entrainant une surconsommation électrique, comme par exemple des timers produisant une activité importante et continue, ou qu'une priorité de processus trop grande soit donnée à l'application.
- **Naming Rules** : ce sont les règles chargées de vérifier le respect des conventions de nommage des différents éléments dans le code. Elles reprennent les conventions édictées dans le ".NET Framework Design Guidelines."
- **Performance Rules** : règles vérifiant que le code est conçu de manière à ne pas avoir un impact potentiellement néfaste sur les performances de l'application.
- **Portability Rules** : règles vérifiant la portabilité possible du code vers différentes plateformes.
- **Security Rules** : règles vérifiant qu'il n'y a pas de problème majeur de sécurité au niveau de la conception du code.
- **Usage Rules** : règles vérifiant que les usages dans le développement d'applications .Net sont respectés. Cela couvre des domaines relativement différents.

5. Ajouter de nouvelles règles personnalisées

FxCop offre la possibilité d'ajouter de nouvelle règles à valider, étendant ainsi quasiment sans limite les vérifications et validations à effectuer sur un code. Dès lors, il devient possible d'ajouter des règles totalement personnalisées, et, collant au plus près aux spécifications et contraintes d'une application. Voilà comment procéder.

Tout d'abord, il faut préciser que les règles sont contenues dans des dll .Net. Il nous faudra donc générer une nouvelle dll pour étendre le nombre de règles prisent en charge par FxCop.

Nous allons utiliser Visual Studio 2005 et le langage Visual Basic .Net pour créer une nouvelle règle, mais sachez que n'importe quel IDE ou même Notepad fera l'affaire, il s'agît juste de créer une dll .Net. Il faut donc choisir le type de projet "Bibliothèque de classe".

Sans plus tarder, il faut ajouter deux références à votre projet. Les références à ajouter concernent deux dll fournies avec FxCop, il s'agît de FxCopSdk.dll et Microsoft.Cci.dll qui se trouvent dans le répertoire d'installation de FxCop, par défaut "C:\Program Files\Microsoft FxCop 1.35"

Le principe est relativement simple, nous allons créer une classe BaseCustomRule qui dérive d'une classe du SDK, la classe BaseIntrospectionRule. Cette nouvelle classe nous servira de classe de base pour les règles que nous voulons ajouter. Cette classe que nous allons créer devra se trouver dans un namespace DeveloppezCustomRules, qui englobera par ailleurs l'ensemble des classes de notre projet. Nous allons aussi ajouter un constructeur qui permet de passer le nom de la règle concernée, une référence au fichier XML, ainsi qu'à l'assembly. Voilà à quoi doit ressembler votre classe de base.

Classe BaseCustomRule

```
Imports System
Imports System.Collections.Generic
Imports System.Text
Imports Microsoft.FxCop.Sdk
Imports Microsoft.FxCop.Sdk.Introspection

Namespace DeveloppezCustomRules
    <CLSCompliant(False)> _
    Public MustInherit Class BaseCustomRule
        Inherits BaseIntrospectionRule
        Public Sub New(ByVal ruleName As String)
               MyBase.New(ruleName,
               "DeveloppezCustomRules.Rules",
               GetType(BaseCustomRule).Assembly)
        End Sub
    End Class
End Namespace
```

Une fois cette classe de base créée, nous pouvons édicter nos règles personnalisées. Il va s'agir de classes dérivant de la classe de base. La première classe que je vais ajouter au projet se nomme DeveloppezRule1. Elle comporte les éléments suivants :

- **Un constructeur** : il prend en paramètre le nom de la règle tel qu'il sera définit dans le fichier XML que nous allons voir plus loin.
- **Une méthode BeforeAnalysis()** qui se substitue à celle de la classe BaseIntrospectionRule. Cette méthode contient le code qui sera exécuté avant de lancer l'analyse de l'assembly, c'est donc le bon endroit, par exemple, pour initialiser des variables ou effectuer je ne sais quelle action qui vous sera utile.
- **Une méthode Check()**, elle substitue à la méthode Check de la classe BaseIntrospectionRule. Cette méthode est la plus importante, c'est elle qui contient le code de l'analyse à proprement parler. Cette fonction retourne une ProblemCollection, qui contiendra comme son nom l'indique, le ou les problèmes rencontrés.
- **Une méthode AfterAnalysis()** qui se substitue à celle de la classe BaseIntrospectionRule. Cette méthode contient le code qui sera exécuté après la vérification de l'assembly par la méthode Check. Dans cette méthode vous pouvez envisager de libérer les ressources utilisées, ou encore, effectuer une quelconque action post-vérification en relation avec votre règle personnalisée.

Cette classe contenant notre règle personnalisée, doit bien évidemment faire partie du namespace DeveloppezCustomRules. Voici à quoi pourrait ressembler votre classe :

Classe DeveloppezRule1

```
Imports Microsoft.Cci
Imports Microsoft.FxCop.Sdk
Imports Microsoft.FxCop.Sdk.Introspection
Namespace DeveloppezCustomRules

    <CLSCompliant(False)> _
    Public Class DeveloppezRule1
        Inherits BaseCustomRule

        Public Sub New()
            MyBase.New("DeveloppezRule1")
        End Sub

        Public Overrides Sub BeforeAnalysis()
        ' Code exécuter avant la                    '
vérification par Check()
        End Sub

        Public Overrides Function Check(
               ByVal aMember As
               Microsoft.Cci.Member) As
        ProblemCollection

            If aMember Is Nothing Then
                Return Nothing
            End If

               If aMember.Name.ToString.
               StartsWith ("FxCop") Then
                      Problems.Add(New Problem
                      (GetResolution(aMember.
                      Name.ToString)))
            End If

            Return Problems

        End Function

        Public Overrides Sub AfterAnalysis()
        ' Code exécuter après la                    '
vérification par Check()

        End Sub
    End Class
End Namespace
```

A la lecture de ces quelques lignes de code je dois vous apporter quelques précisions, notamment à propos de la méthode check. Cette méthode propose six surcharges et donc peut prendre des types de paramètres différents. En voici la liste :

- namespaceName As **String**, type As **Microsoft.Cci.TypeNodeLists**
- module As **Microsoft.Cci.Module**
- resource As **Microsoft.Cci.Resource**
- type As **Microsoft.Cci.TypeNode**
- member As **Microsoft.Cci.Member**

- parameter As **Microsoft.Cci.Parameter**

Ici, elle prend en paramètres un membre de type Microsoft.Cci.Member, c'est le cas puisque notre règle va porter sur l'analyse des membres d'une classe. Son code, que j'ai recopié ci-dessous, nous montre que la vérification va porter le nom des membres de l'assembly, et la condition de "validation" va être que ce nom ne commence pas par "FxCop". Si jamais c'est le cas, alors on retourne le problème dans une ProblemCollection en utilisant la méthode Add, qui en paramètre prend un objet Problem. Cet objet Problem va lui, prendre en paramètre, la "solution", en effet nous allons pouvoir ici passer une liste de chaînes que l'on pourra retourner à l'utilisateur dans le rapport d'analyse. Vous comprendrez plus tard à quel point ce détail est pratique. Ici, je me contente de retourner le nom de la méthode; cela me permettra de l'afficher dans le détail du problème que FxCop aura détecté par le biais de notre règle personnalisée. Je me contente de retourner seulement le nom du membre concerné par le problème, mais je peux potentiellement faire passer n'importe quelle information, du moment qu'elle est de type System.String.

Fonction Check()

```
Public Overrides Function Check(ByVal aMember As
        Microsoft.Cci.Member) As ProblemCollection
        If aMember Is Nothing Then
                Return Nothing
        End If

        If      aMember.Name.ToString.StartsWith("FxCop")
Then
                Problems.Add(New Problem(
                GetResolution(aMember.Name.ToString)))
        End If

        Return Problems
End Function
```

Nous voyons ici que nous utilisons des tests relativement simples, à base de If / End If, mais, sachez qu'il est possible de faire des choses beaucoup plus complexes, et que l'analyse peut être très poussée, on le constate d'ailleurs avec les règles déjà existantes. Je tiens à attirer votre attention sur l'énumération "Microsoft.cci.NodeType" qui permet de déterminer quel élément il faut cibler dans l'assembly. Cette énumération est très pratique et vous l'utiliserez quasiment à chaque fois.

Fichiers Rules.xml

```
<?xml version="1.0" encoding="utf-8" ?>
        <Rules FriendlyName="Règles Developpez.com">
                <Rule TypeName="DeveloppezRule1"
                Category="Microsoft.Naming"
        CheckId="DVP001">
        <Name>DeveloppezRule1   :   noms   de   membres
        commencant par "FxCop"</Name>
        <Description>Les noms de membres ne doivent pas
        commencer par "FxCop"</Description>
        <GroupOwner>Developpez.com</GroupOwner>
        <DevOwner>Webman</DevOwner>
        <Owner>Webman</Owner>
        <Url>http://webman.developpez.com</Url>
       <Resolution>Modifier le nom du membre "{0}" de
        manière à ce qu'il ne commence pas par "
        FxCop"</Resolution>
        <Email>mail@mail.com</Email>
        <MessageLevel Certainty="100">Warning
        </MessageLevel>
        <FixCategories>Breaking</FixCategories>
  </Rule>
</Rules>
```

Entrons un peu dans le détail :

- **TypeName** : c'est le nom de la règle, celui que l'on passe au constructeur de la règle personnalisée. Il doit être unique.
- **Category** : c'est la catégorie de la règle, ici elle correspond à la règle de nommage.
- **CheckID** : est un identifiant unique de la règle.
- **Name** : c'est le nom de la règle.
- **Description** : comme son nom ne l'indique absolument pas... il s'agît de la description de la règle.
- **GroupOwner** : on spécifie généralement ici le nom de la société.
- **Owner** : le nom du développeur qui à créer cette règle.
- **Url** : on spécifie ici une URL pointant vers une aide en ligne correspondant à l'erreur associée à la règle.
- **Resolution** : c'est le message qui apparait dans le détail de l'anomalie détectée, il propose une solution. C'est dans ce message que l'on peut afficher les informations passées en paramètres à la méthode GetResolution(). Pour afficher ces informations, il suffit juste de mettre entre accolades {} l'index de la chaine dans la liste de paramètre. Attention, l'index commence à {0} et non à {1} ; de plus s'il l'on spécifie un index hors tableau une exception sera levée, il faut donc être attentif à ce que l'on fait.
- **Email** : l'adresse mail du développeur ou de l'entreprise auteur de la règle.
- **MessageLevel** : l'importance du problème détecté. Cela peut être une des six catégories suivantes : Error, Critical Error, Warning, Critical Warning, Information.
- **MessageLevel** Certainty : le pourcentage de probabilité que l'anomalie détectée en soit réellement une, généralement on utilise un chiffre entre 1 et 99%, mais l'on peut également utiliser 100% si on le souhaite.
- **FixCategories** : la valeur peut être Breaking, NonBreaking ou Unknown, cela signifie respectivement que si l'on applique la solution proposée, il faudra aussi recompiler les assemblies ayant des dépendances avec l'assembly comportant l'anomalie, sous peine d'être confronté à des erreurs d'exécution.
- **FriendlyName** : ce nom ne concerne pas une règle en particulier, mais l'ensemble des règles contenues dans la dll, ce nom s'affiche dans l'interface de FxCop, dans la liste des dll des règles chargées.

Remarque : le fichier XML devra être intégré dans la dll, cela signifie que "Actions de génération" devra être positionné sur "*Ressource incorporée*", et que "Copier dans le répertoire de sortie" devra être positionné à "*Ne pas copier*".

A ce stade, vous n'avez plus qu'à générer votre projet. La dll est alors créée, il vous faut aller dans FxCop, puis dans le menu "Project", "Add Rules" et sélectionner la dll fraichement générée. Vos nouvelles règles apparaissent alors dans la liste, et vous pouvez activer ou non vos règles en globalité, ou alors règle par règle.

Pour comprendre mieux le processus de création de règles personnalisées, vous avez accès à deux autres règles personnalisées sur la version en ligne de cet article : Lien33

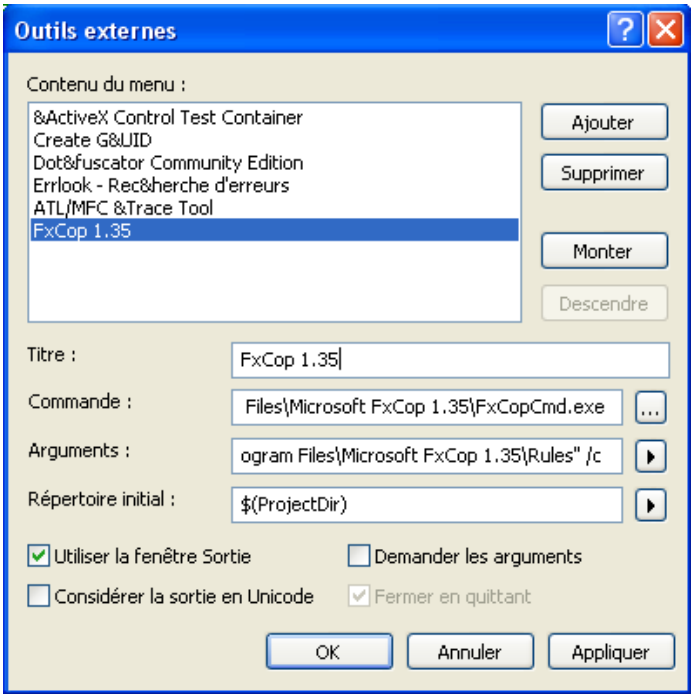
6. Intégrer FxCop dans Visual Studio 2005

FxCop est un outil autonome, et indépendant de Visual Studio

2005 (sauf dans les versions Team System et Team Suite pour lesquelles il est directement intégré dans l'IDE) mais il pourrait être intéressant de l'intégrer en tant "qu'outil externe". Cela est tout à fait réalisable, voici comment procéder.

- Aller dans menu "Outils", "Outils externes", bouton "Ajouter".
- Dans le titre, entrez par exemple "FxCop 1.35"
- Dans commande : entrez le chemin pointant vers l'exécutable "FxCopCmd.exe" qui se trouve dans le répertoire d'installation de FxCop, sur ma machine, le chemin complet est "C:\Program Files\Microsoft FxCop 1.35\FxCopCmd.exe".
- Dans arguments : entrez la ligne suivante, adaptée si besoin au chemin du répertoire " Rules " si le vôtre est différent. "/f:$(TargetPath) /r:"C:\Program Files\Microsoft FxCop 1.35\Rules" /c"
- Dans répertoire initial, entrez "$(ProjectDir)"
- Cochez "Utiliser la fenêtre de sortie", ainsi le résultat de l'analyse s'affichera dans cette même fenêtre.

Voici à quoi devrait ressembler la fenêtre de configuration des outils externes :



Maintenant vous devez fermer et ré-ouvrir Visual Studio 2005. L'avantage ici, est que FxCop est directement intégré dans l'IDE, et accessible depuis le menu "Outil", mais personnellement je n'aime pas cette intégration, car afficher le rapport dans la fenêtre de sortie n'est pas très exploitable... En tout cas, bien loin de l'ergonomie de l'interface de FxCop. C'est donc pratique de pouvoir accéder à FxCop de cette manière, mais cela ne dispensera pas d'utiliser FxCop en tant qu'application indépendante.

Dans l'invite de commande de Visual Studio 2005 (ou du SDK), tapez ce qui suit :

7. Conclusion

Au cours de cet article nous avons vu que FxCop est un outil d'analyse de code très complet, possédant plus de 200 règles, ce qui couvre un domaine vaste. Si vous suivez les recommandations, vous êtes à peu près certain d'avoir un code conforme aux bonnes pratiques .Net, et vous éviterez les problèmes de sécurité majeurs. Mais, un tel outil ne pouvant pas être exhaustif, nous avons vu en contre-partie qu'il est relativement aisé de créer ses propres règles personnalisées d'analyse grâce à un SDK, qui présente malheureusement l'énorme inconvénient de ne pas être documenté, espérons que ce point évoluera rapidement. En effet, ces fonctionnalités sont très intéressantes et mérient que l'on s'y intéresse. J'espère, au travers de cet article, avoir réussi à vous montrer l'utilité réelle, et le gain de productivité que peut vous apporter FxCop, et que désormais, cet outil fera partie des applications que vous utiliserez au quotidien.

Retrouvez la suite de l'article de Ronald Vasseur : Lien33

Extraits d'articles

Reporting avec Visual Studio 2005

Tous les états de ce nouvel outil ne sont ni plus ni moins que des fichiers XML. Et oui, encore du XML, comme les documents Office et autres. Vous voyez sans doute où je veux en venir. Le XML nous offre la possibilité de pouvoir éditer ces fichiers d'état ailleurs que dans Visual Studio. Personnellement, possédant Visual Studio, je vais en profiter. Vous comprendrez pourquoi en examinant ce fichier qui n'est autre que l'état que nous allons créer dans la partie 2. Convaincu ?

Découvrez le reporting avec Visual Studio 2005 d'Olivier Delmotte : Lien34

Création et mise en forme d'états Crystal Reports avec VS .NET

Cet article explique comment créer et mettre en forme un état Crystal Reports simple avec Visual Studio .NET.

Un article similaire rédigé par David Pédehourcq vous présente plus en détails le fonctionnement d'un état et le déploiement de ce dernier dans une application windowsform. L'objectif de mon article est de présenter une manière légérement différente de remplir un état et d'aller un peu plus loin dans sa mise en forme.

Les codes présentés dans cet article ont été conçu avec Visual Studio .NET 2003

Apprenez à créer et à mettre en forme des états Crystal Reports avec Manuel Sergent : Lien35

Vu sur les Blogs

Developer & IT Pro Days 2007 (Gand - Belgique)

Developer & IT Pro Days 2007 (Gand - Belgique)

C'est enfin annoncé. Vu que c'est officiel, je vais enfin pouvoir faire de même et vous annoncer ces dates:

Il s'agit du 28 et 29 mars 2007.

Actuellement on n'en sait pas beaucoup, mais voici déjà les infos importants :

```
Join us on March 28-29, 2007 in Ghent at the International Convention Center Ghent (ICC).
Sign the Developer & IT Pro Days 2007 Guestbook.
Be among the first to know the latest about Developer & IT Pro Days 2007 in Ghent. When you're in the
Guestbook, you're invited as soon as registration is open so you can plan, prepare, and make the most of your
experience.

Developer & IT Pro Days 2007 Pricing & Registration
Registration will go live soon.
Registration rate is €299 for both conference days and $399 including pre-conference day.

Conference Agenda

Tuesday 27 March : Pre-Conference Seminars
Wednesday 28 March : Day 1
Thursday 29 March : Day 2
```

Retrouvez le billet de Didier Danse en ligne : Lien36

Les derniers tutoriels et articles

Utilisation de Lua comme langage de Script : Utiliser Lua dans du code C

Dans ce tutoriel, je présente l'utilisation du langage de script Lua dans un programme C. Ce tutoriel n'a pas pour but d'expliquer la syntaxe et la programmation en Lua. L'objectif de ce tutoriel est de montrer comment interfacer Lua et le C pour qu'ils puissent communiquer ensemble.

1. Pourquoi utiliser un langage de scripts ?

De nombreux moteurs de jeux utilisent un langage de scripts à plusieurs niveaux du déroulement du programme. On peut se demander quels sont les avantages à utiliser un langage de scripts au lieu de tout programmer directement en C dans son moteur de jeu. Ces avantages sont multiples, en voici quelques uns.

Un problème apparaît quand un projet commence à prendre de l'ampleur : les temps de compilations peuvent devenir de plus en plus importants, allant de plusieurs minutes lorsqu'on ne recompile qu'une partie du projet à plusieurs dizaines de minutes, voire même plusieurs heures pour les très gros projets, lorsqu'on doit recompiler tout le projet. Ces temps de compilation font perdre un temps précieux, surtout lorsqu'on est en phase de mise au point et que le changement d'une variable fait perdre dix minutes... Sur ce point, les langages de scripts on l'avantage d'être évalué à l'exécution, il n'y a donc pas de temps de compilation, et on gagne donc du temps pour tous les problèmes de mise au point.

Un autre problème est que les développeurs de jeux vidéo ne sont pas tous des programmeurs. Il y a des graphistes, des level designers et autres qui peuvent être appelés à devoir changer le comportement du jeu pour mieux correspondre aux objectifs visés. N'ayant pas forcément de notions avancées de la programmation, il faut pouvoir leur donner la possibilité d'effectuer ces changements sans pour autant savoir ce qu'est un pointeur, une allocation mémoire ou autre. Les langages de scripts ont donc généralement une syntaxe simple et se limitent à des paradigmes de programmation connus, permettant ainsi une programmation plus simple à appréhender. De plus, ces langages gèrent généralement la mémoire eux même avec, par exemple, un ramasse-miettes (garbage collector en anglais) pour le langage que nous allons étudier ici.

Un autre point qui peut pousser à l'utilisation de langages de scripts vient des mods. En effet, on peut vouloir proposer à l'utilisateur de faire ses propres modifications du jeu sans pour autant lui donner tout le code source. Pour cela, on peut mettre en place des systèmes basés sur l'utilisation de bibliothèques dynamiques comme dans les derniers moteurs d'ID software, mais on peut aussi réaliser un système ou le moteur du jeu est dirigé par des scripts facilement modifiables par l'utilisateur. L'avantage des scripts par rapport aux DLL est qu'il n'y a pas besoin d'avoir un compilateur sous la main pour pouvoir réaliser son mod. Ceci permet une plus grande ouverture aux utilisateurs.

Malgré ces avantages, l'utilisation de scripts a aussi des inconvénients. Les scripts sont plus lents que du code compilé, ce qui oblige à bien choisir quelles parties du programme seront réalisées en code natif et quelles seront celles réalisées dans le langage de scripts. Ainsi, on réalisera certainement les algorithmes de recherches de chemins en C, mais on donnera la possibilité à l'utilisateur d'exploiter les résultats dans les scripts.

Un autre problème est que les scripts sont généralement stockés en mode texte directement lisible par l'utilisateur. Si ceci permet une modification plus facile, ça permet aussi aux petits malins de changer le comportement du jeu pour réaliser des exploits ou prendre l'avantage durant une partie réseau. Néanmoins, certains langages de scripts permettent de pré- compiler les scripts, comme c'est le cas de celui que nous allons utiliser.

2. Description de Lua

Le langage de scripts que nous allons utiliser est Lua. Ce langage de scripts à été crée en 1991, et a déjà été utilisé dans de nombreux jeux commerciaux comme par exemple dans Far Cry de Crytek. Je ne présente pas plus l'historique de Lua dans ce tutoriel, pour cela, vous pouvez toujours aller sur le site officiel, mais je vais rapidement présenter quelques points importants sur le fonctionnement du langage.

Ce langage a l'avantage de pouvoir utiliser indifféremment des scripts en mode texte ou des scripts compilés. Les scripts compilés en Lua ne sont plus lisibles par l'utilisateur, et permettent un temps de chargement plus rapide, mais, par contre, l'exécution n'est pas plus rapide étant donné que c'est le même moteur de scripts qui tourne par derrière. En fait, la compilation des scripts ne fait que créer la structure du programme directement pour que Lua puisse l'utiliser, et sauvegarde le résultat dans un fichier. Ainsi, lors du chargement du script compilé, Lua n'a plus à faire l'analyse et peut donc immédiatement l'utiliser, ce qui accélère les temps de chargements.

Lua utilise une pile d'exécution virtuelle pour fonctionner. Cette notion est très importante car l'ensemble de l'interfaçage entre Lua et le C passe par cette pile virtuelle. Ceci est intéressant notamment pour le passage de paramètres aux fonctions Lua et pour récupérer les valeurs de retour. Il est important lorsqu'on développe un programme qui utilise Lua de bien faire attention à ce que la pile ne passe pas dans un état incohérent pour éviter d'avoir des comportements indéterminés de l'interpréteur de scripts. En fait c'est la même chose que pour un programme en assembleur : une mauvaise utilisation de la pile peut avoir des conséquences invisibles au départ et donc très difficiles à débuguer par la suite... La technique pour toujours avoir une pile valide est de toujours remettre la pile dans l'état dans lequel on l'a

trouvée au début de notre code, ainsi, on revient à chaque fois à un état cohérent.

Il est important de noter que lorsqu'on fait appel à Lua dans notre code C, on utilise intensivement la pile via des indices.
Ces indices sont gérés de deux façons différentes :

- Les indices positifs sont des indices relatifs à la base de la pile.
- Les indices négatifs sont des indices relatifs au sommet de la pile.

Le langage propose aussi un mécanisme d'extension du fonctionnement du langage via un système de meta table. Ce système est relativement complexe, je ne l'expliquerai donc pas ici, mais nous l'utiliserons pour mettre en place un système de programmation objet qui n'est pas supporté de base par Lua. En effet, à la base, Lua est un langage de programmation procédurale, on ne peut donc utiliser que des fonctions comme en C. C'est cette programmation procédurale que je présente dans ce tutoriel.

L'installation de Lua est très simple, elle consiste en un fichier de bibliothèque (.lib sous visual studio, .a sous gcc) et en quelques headers qu'on peut soit copier dans le répertoire include du compilateur, soit mettre dans le projet et signaler au compilateur d'aller les chercher (mais la je vous laisse faire, c'est à vous de savoir configurer votre compilateur ;-)).
Les fichiers nécessaires à l'utilisation de Lua sont fournis dans le fichier zip du tutoriel que vous trouverez en bas de la page.

Pour pouvoir utiliser Lua, il ne nous reste plus qu'à inclure les fichiers lua.h, lualib.h et lauxlib.h si on est en C, et le fichier lua.hpp si on est en C++. Voila, on peut maintenant commencer à utiliser Lua.

Avant de détailler plus l'interfaçage de Lua avec du C, il faut d'abord savoir comment créer un contexte d'exécution Lua et lancer un script. Ceci se fait comme cela :

Initialisation de Lua
```
lua_State * state;
// on créer un contexte d'exécution de Lua
state = lua_open();
// on charge les bibliothèques standards de Lua
luaL_openlibs(state);
// on lance le script lua
if (luaL_dofile(state,"script.lua")!=0){
	// il y a eu une erreur dans le script
	fprintf(stderr,"%s\n",lua_tostring(state,-1));
	exit(0);
}
```

Dans ce code, la variable state représente le contexte d'exécution de Lua. Cette variable est très importante car elle sera utilisée lors de chaque appel de fonction Lua pour déterminer sur quel contexte travailler. En effet, on peut très bien avoir plusieurs scripts Lua lancé en même temps avec chacun sa pile d'exécution distincte des autres.

Un autre point important de Lua est que les variables ne sont pas typées. C'est le contenu des variables qui détermine son type. Ainsi, une même variable pourra être coup sur coup un nombre puis une chaîne de caractères. Ceci nous obligera donc dans nos programmes à vérifier que les variables Lua qu'on récupère sont bien du bon type. Il existe en Lua un type particulier : nil. Une variable typée nil est en fait une variable qui n'a pas de type (donc pas de valeur assignée).

3. Utiliser les variables entre Lua et son programme C

Dans cette partie, je montre comment faire pour récupérer les valeurs de variables Lua en C. Je détaille aussi la lecture des tables Lua depuis le C car ce point est un peu plus complexe qu'une simple lecture de variable.

3.1. Lire une variable Lua

C'est, historiquement, le premier modèle d'éclairage en temps réel utilisé. En effet, l'éclairage en temps réel étant coûteux, lorsque les ordinateurs n'avaient pas de carte accélératrice, il fallait utiliser un système très rapide, même si il présente les désavantages présenté précédemment.

C'est ce modèle dont nous allons étudier la mise en oeuvre dans ce premier tutoriel.

La lecture d'une variable Lua est très simple. Il suffit de demander à Lua d'empiler le contenu de la variable en connaissant son nom. Lua met la valeur en sommet de pile, il ne nous reste plus qu'à vérifier que la valeur empilée est bien du type souhaité et à récupérer la valeur. Voici par exemple le code nécessaire pour récupérer un nombre.

Code de récupération d'un nombre en Lua depuis le C
```
lua_settop(state,0);
lua_getglobal(state,"var");
if (lua_isnumber(state,1)){
	printf("valeur de la variable var :
%f\n",lua_tonumber(state,1));
}
lua_pop(state,1);
```

Et voici le code pour récupérer une string.

Code de récupération d'une string en Lua depuis le C
```
lua_settop(state,0);
lua_getglobal(state,"toto");
if (lua_isstring(state,1)){
	printf("valeur de la variable toto :
	%s\n",lua_tostring(state,1));
}
lua_pop(state,1);
```

Il est important de bien dépiler la valeur après l'avoir récupérée, car sinon, on risque de mettre la pile dans un état incohérent.

Comme vous pouvez le voir, ce code est relativement simple et on peut très bien en faire une petite fonction qui va nous faciliter le travail, mais je vous laisse la faire ;-)

3.2. Lire une table Lua

La lecture de table est un peu plus compliquée que pour une simple variable. En Lua, une table peut être vu comme un tableau indexé par n'importe quel type de variable. Ainsi, on peut très bien indexer une table en même temps avec des entiers et des chaînes de caractères. Du coup, la lecture de la table est plus complexe.

Il faut d'abord demander à Lua d'empiler le tableau en sommet de pile. Ensuite, on empile la valeur de l'indice du tableau qu'on souhaite lire, puis on demande à Lua de remplacer cet indice par sa valeur. On peut ensuite lire la valeur comme une variable classique. Voici un exemple de code pour lire la valeur de la table Lua tableau[2] :

Récupération d'une valeur dans une table Lua depuis C

```
//récupération du contenu d'un tableau Lua depuis le code C
lua_settop(state,0);
lua_getglobal(state,"tableau");
if (!lua_istable(state,1))
{
	fprintf(stderr,"la variable tableau n'est pas un tableau\n");
	lua_pop(state,1);
}
else
{
	// on veut adresser la variable d'indice 2
	lua_pushnumber(state,2);
	// maintnenant on a dans la pile l'indice à la place 1 et le tableau
	// à la place 2 (car on a empilé l'indice par dessus)
	// on demande à Lua de remplacer le haut de la pile (donc notre indice)
	// par le contenu de la case du tableau en lui donnant l'adresse du tableau
	lua_gettable(state,-2);
	// on vérifie la donnée recuperée
	if (lua_isstring(state,-1))
	{
		printf("valeur recuperé à l'indice 2 : %s\n",lua_tostring(state,-1));
	}
	// on dépile les deux éléments (la valeur de la case et le nom
	// du tableau)
	lua_pop(state,2);
}
```

Encore une fois, on s'assure que la pile retourne dans un état cohérent après la lecture.

3.3. Passer une variable à Lua

Contrairement à ce qu'on pourrait penser, ici, je ne montre pas la méthode à mettre en oeuvre pour passer une variable C à Lua... En fait, ceci est une mise en garde. Si on peut facilement lire une variable Lua depuis le C et vice versa, il ne faut pas oublier que les deux variables ont une vie différente. La variable C va évoluer dans le contexte de l'application C alors que la variable Lua va évoluer dans le contexte du script Lua... Le passage de variables C vers Lua doit donc être évité autant que possible, et ne doit être utilisé que pour des passages de paramètres aux fonctions Lua.

4. Utiliser des fonctions entre Lua et son programme C

Bien que le fait de pouvoir lire des variables Lua depuis le C puisse être très pratique, pour utiliser les fichiers de scripts comme des fichiers de configuration par exemple, on se retrouve très vite limité. En effet, avec juste des variables, on ne peut pas faire d'interfaçage complexe entre notre programme et nos scripts. Pour cela, il faut pouvoir utiliser du code C dans Lua et du code Lua dans notre code C. C'est ce que nous allons voir maintenant. Dans un premier temps, je montre comment appeler du code C depuis Lua, puis je montre comment appeler des fonctions Lua depuis notre code C.

4.1 Appeler une fonction C depuis Lua

Le fait de pouvoir appeler des fonctions C depuis notre code Lua est ce qui nous permet de diriger notre programme C via Lua. Pour cela, c'est très simple. Il suffit d'avoir une fonction qui a la signature suivante.

Signature d'une fonction C appelable depuis Lua

```
int functionName(lua_State* L);
```

Le type de retour doit être un entier qui spécifie le nombre de valeurs de retour (Lua permettant de retourner plusieurs valeurs), et le paramètre de la fonction est le contexte d'exécution dans lequel est appelée la fonction. Voici un exemple de fonction C simple qu'on va appeler depuis Lua :

Fonction helloWorld appelée depuis Lua

```
int helloWorld(lua_State* L){
	printf("hello world/n");
	return 0;
}
```

Remarquez que la fonction ne retournant pas de valeurs à Lua, notre fonction C retourne 0.

Pour pouvoir appeler cette fonction, il faut aussi signaler à Lua son existence et lui donner son adresse pour que Lua puisse l'appeler, ce qui se fait comme ceci :

Enregistrement de notre fonction C pour l'appeler depuis Lua

```
lua_register(state,"helloWorld",helloWorld);
```

Le premier paramètre est le contexte d'exécution du script, le second est le nom de la fonction dans Lua (on peut très bien spécifier un nom différent dans Lua que celui utilisé en C), et le dernier paramètre est la fonction à appeler par Lua.

Maintenant nous pouvons appeler la fonction dans un script Lua, elle est bien reconnue et exécutée.

Nous savons maintenant appeler une fonction Lua, mais nous n'avons pas vu comment récupérer les paramètres de cette fonction. C'est ce que nous allons voir maintenant.

Dans Lua, les paramètres sont passés dans la pile d'exécution, ils sont donc empilés et nous les retrouvons en haut de pile au début de notre fonction C. En fait, lors de l'appel d'une fonction, Lua empile les paramètres dans l'ordre dans lequel ils apparaissent dans le script, puis il déplace la base de la pile pour qu'elle se retrouve au niveau du premier paramètre. Nous avons donc le premier paramètre à l'indice 1 (Lua commence à compter à partir de 1), le second à l'indice 2 etc... Ceci permet de s'assurer qu'une fonction ne peut pas modifier la pile de la fonction qui l'a appelée. Pour récupérer le nombre d'arguments, il faut appeler la fonction lua_gettop(), et ensuite, on peut récupérer chaque argument en fonction de sa position relative à la base de la pile grâce au fonction lua_to* comme illustré dans les fonctions suivantes.

Fonction C affichant une chaîne de caractères passée depuis Lua

```
int affichage(lua_State* L){
	int nbArguments = lua_gettop(L);
	if (nbArguments != 1){
		fprintf(stderr,"nombre d'arguments invalide");
		return 0;
	}
	printf("parametre : %s\n",lua_tostring(L,1));
	return 0;
}
```

La première fonction prend en paramètre une chaîne de caractères et l'affiche dans la console. On peut remarquer que c'est au programmeur d'effectuer la vérification que les paramètres passés sont valides.

Fonction C affichant tous les paramètres passés par Lua

```
int printParams(lua_State* L){
	printf("affichage des paramètres de la fonction
	: \n");;
	int nbParams = lua_gettop(L);
	// Attention : lua commence à compter à partir
	de 1
	for (int i = 1; i <= nbParams; i++){
		if (lua_isnumber(L,i)){
			printf("%f\n",lua_tonumber(L,i));
		} else if (lua_isboolean(L,i)){
			if (lua_toboolean(L,i)){
				printf("true\n");;
			} else{
				printf("false\n");
			}
		} else if (lua_isstring(L,i)){
			printf("%s\n",lua_tostring(L,i));
		}
	}
	return 0;
}
```

La deuxième fonction affiche aussi les paramètres qui lui sont passés, mais cette fois le nombre de paramètres n'est pas fixé, et leur type non plus.

Notez que les indices dans la pile en Lua commencent à 1 et non pas à 0 comme en C. Pour adresser la base de la pile il faut donc adresser la variable d'indice 1.

Encore une fois, ces fonctions ne retournent pas de valeurs de retour à Lua, elles retournent donc 0. Nous allons maintenant voir comment faire pour retourner des valeurs de retour à Lua avec la fonction suivante.

Cette fonction prend un nombre en entrée et retourne le nombre divisé par deux. Comme vous pouvez le voir, l'envoi de valeurs de retour à Lua est très simple, il suffit d'empiler la valeur en sommet de pile et de signaler que la fonction retourne bien des valeurs de retour en retournant le nombre de valeurs empilées. Voici donc la fameuse fonction :

Fonction montrant l'utilisation de la pile pour envoyer à Lua

```
int halfParam(lua_State* L){
	int nbParam = lua_gettop(L);
	if (nbParam != 1){
		printf("halfParam : mauvais nombre de
		paramètres\n");
		return 0;
	}
	if (!lua_isnumber(L,1)){
		printf("halfParam : le paramètre doit
		etre de type number\n");
		return 0;
	}
	float param = lua_tonumber(L,1);
	float ret = param / 2.0f;
	lua_pushnumber(L,ret);
	return 1;
}
```

Bien entendu, pour que ces fonctions puissent être appelées depuis Lua, il ne faut pas oublier de les enregistrer auprès de Lua comme ceci :

```
lua_register(state,"affiche",affichage);
lua_register(state,"printParams",printParams);
lua_register(state,"halfParam",halfParam);
```

Maintenant que nous savons comment appeler du code C depuis Lua, il nous faut voir comment faire pour appeler du code Lua depuis le C.

4.2. Appeler une fonction Lua depuis le C

Bien que la majeure partie de l'interface entre le C et Lua consiste à créer des fonctions C qui seront appelées depuis Lua, il peut être pratique d'appeler des fonctions Lua pour plusieurs raisons :

- Créer une fonction callback qui se déclenche sur un événement donné du moteur de jeu (un ennemi repère le joueur, le joueur appuie sur un bouton...).
- Appeler une fonction qui peut être surchargée par l'utilisateur. Lorsqu'une fonction définie en C est redéfinie dans un script Lua, c'est cette dernière qui sera appelée lors du déroulement du script. On peut donc vouloir appeler depuis le C cette fonction surchargée au lieu d'utiliser directement la fonction C.
-

Pour utiliser des fonctions Lua, nous verrons d'abord comment vérifier l'existence d'une fonction, puis nous verrons comment appeler une fonction simple, ensuite, je montrerai comment appeler des fonctions avec paramètres, puis les fonctions avec valeurs de retour.

4.2.1. Vérifier l'existence de la fonction Lua

Pour pouvoir appeler une fonction Lua depuis notre code C, il faut d'abord vérifier son existence. Pour cela, il faut empiler le nom de la fonction que l'on souhaite appeler, puis faire appel à lua_isfunction() avec comme paramètres le contexte d'exécution Lua et l'indice relatif au sommet de la pile ou a été empilé le nom de la fonction à tester. Notez bien ici que l'indice soit un indice négatif. Ceci signifie à Lua que l'indice est relatif au sommet de pile. Dans Lua, les indice positif son relatifs à la base de la pile, et les indice négatif sont relatifs au sommet de la pile. Le code pour vérifier l'existence d'une fonction donne donc ça :

```
// appel d'une fonction Lua depuis le code C
// on empile le nom de la fonction qu'on souhaite
lancer
lua_getglobal(state,"funcName");
// on vérifie si la fonction existe bien
if (!lua_isfunction(state,-1)){
	// la fonction n'existe pas
	fprintf(stderr,"la fonction funcName n'existe
pas\n");
	lua_pop(state,1);
} else{
	// code à effectuer pour l'appel de la fonction
}
```

4.2.2. Fonction simple

Maintenant que nous savons comment faire pour vérifier que nous pouvons bien appeler une fonction, il ne nous reste plus qu'à appeler cette fonction. Pour cela rien de plus simple, il suffit :

- d'empiler le nom de la fonction qu'on souhaite appeler. C'est ce que nous avons fait lors de la vérification de l'existence de la fonction dans Lua.
- d'utiliser la fonction lua_call.

La fonction lua_call prend en paramètre le contexte d'exécution de Lua, le nombre d'arguments de la fonction et le nombre de valeurs de retour. Dans le cas d'une fonction simple sans paramètres et sans valeurs de retour, le code d'appel complet donne donc :

Appel d'une fonction Lua depuis le C

```
// on empile le nom de la fonction qu'on souhaite lancer
lua_getglobal(state,"helloWorld");
// on vérifie si la fonction existe bien
if (!lua_isfunction(state,-1)){
	// la fonction n'existe pas
	fprintf(stderr,"la fonction helloWorld n'existe pas\n");
	lua_pop(state,1);
} else{
	// on appelle la fonction qui a 0 argument et 0 retour
	lua_call(state,0,0);
}
```

4.2.3. Fonction avec paramètres

Pour le cas des fonctions avec paramètres, il suffit de rajouter, entre l'empilement du nom de la fonction et l'appel à lua_call, l'empilement des paramètres dans l'ordre que nous avons vu plus haut. Il ne faut pas oublier non plus de bien donner le bon nombre de paramètres lors de l'appel à lua_call. Voici un exemple simple d'appel de fonction prenant en paramètre une chaîne de caractères :

Appel d'une fonction Lua depuis le code C avec un paramètre

```
// on empile le nom de la fonction qu'on souhaite lancer
lua_getglobal(state,"luaPrint");
// on vérifie si la fonction existe bien
if (!lua_isfunction(state,-1)){
	// la fonction n'existe pas
	fprintf(stderr,"la fonction luaPrint n'existe pas\n");
	lua_pop(state,1);
} else {
	// la fonction existe, on lui passe une chaine de caracteres comme
	// premier argument
	lua_pushstring(state,"le message passé en paramètre");
	// on appel la fonction qui a 1 argument et 0 retour
	lua_call(state,1,0);
}
```

4.2.4. Fonction avec valeur de retour

Maintenant, il ne nous reste plus qu'à savoir récupérer la valeur de retour d'une fonction Lua pour pouvoir l'exploiter dans nos programmes C. Comme pour tout en Lua, il faut passer par la pile. Comme nous l'avons vu, dans la partie sur les retours dans des fonctions C, les valeurs de retour sont empilées en sommet de pile. Pour les récupérer, il nous suffit donc de lire les valeurs en sommet de pile avec un indice négatif, et voila, nous avons nos valeurs de retour... Voila un petit exemple d'appel à une fonction qui effectue l'addition entre deux nombres passés en paramètres et retourne le résultat de cette addition :

Appel d'une fonction Lua et récupération de la valeur de retour

```
// on empile le nom de la fonction qu'on souhaite lancer
lua_getglobal(state,"addition");
// on vérifie si la fonction existe bien
if (!lua_isfunction(state,-1)) {
	// la fonction n'existe pas
	fprintf(stderr,"la fonction addition n'existe pas\n");
	lua_pop(state,1);
} else {
	// la fonction existe, on lui passe une chaine de caractères comme
	// premier argument
	lua_pushnumber(state,10);
	lua_pushnumber(state,20);
	// on appel la fonction qui a 2 argument et 1 retour
	lua_call(state,2,1);
	// on récupère la valeur de retour
	float retour = (float)lua_tonumber(state,-1);
	printf("valeur de retour : %f\n",retour);
}
```

5. Conclusion

Comme vous pouvez le voir, l'utilisation de Lua en C est relativement simple. Nous avons maintenant tous les outils en mains pour pouvoir faire communiquer ces deux langages sans difficultés. Néanmoins, nous ne pouvons pas utiliser de classes en Lua dans l'état actuel des choses. Pour cela, il nous faut utiliser les meta tables de Lua. C'est ce que nous verrons dans le prochain tutoriel sur Lua.

Retrouvez l'article de Michel de Verdelhan en ligne :Lien37

Livres C/C++

C précis et concis De Peter Prinz et Ulla Kirch-Prinz

Critique par Nicolas Joseph

Voici un livre au format poche bien pensé et très pratique.
Le livre de référence du langage C (le K&R) n'est pas épais mais il peut être tout de même fastidieux de trouver une information précise dedans : par exemple si vous cherchez le tableau sur la priorité des opérateurs, l'index référence cinq pages alors qu'avec ce livre il n'y en a qu'une !
En plus de ce côté pratique, il est très complet : tout le langage C est passé en revue (syntaxe, bibliothèque standard et surtout la norme C99), claire : l'auteur n'hésite pas à prendre quelques lignes pour expliquer clairement chaque point du langage et l'abondance des tableaux récapitulatifs facilite la lecture.
Pour finir un petit point qui peut sembler insignifiant mais qui m'a fait adopter ce livre : toutes les nouveautés du C99 sont regroupées en index !
Et tout ça pour un prix réduit. Seul inconvénient : tout le monde va vouloir vous l'emprunter :D

Retrouvez tous les livres C et C++ : Lien38

Visual Basic 6.0 et le format XML

Le format de données XML se répand de plus en plus dans le monde informatique aussi bien sur les réseaux intranet que sur le Web lui-même. Un tour d'horizon très rapide et non-exhaustif des avantages de cette technologie d'échange de données, exploitée via Visual Basic 6.0 est le but de ce petit tutoriel.

1. Quelques notions

1.2. Le XML c'est quoi ?

C'est un langage qui permet de décrire une classe d'objets "documents xml". Au sein de ces documents, le XML impose une structure aux données de façon lisible dans un simple format texte. Le XML utilise le principe bien connu de balises mais celles-ci sont propres au créateur du document et non pas standardisées comme pour du HTML. Il est donc évidemment impératif de bien choisir ses balises pour la bonne compréhension de la structure du document. Un exemple est bien plus parlant :

```
 <?xml version="1.0" encoding="ISO-8859-1" ?>
<developpez>
   <membre fonction="Responsable Visual Basic">
      <nom>Cécile Muno</nom>
      <pseudo>khany</pseudo>
      <activite>
         <date-inscription>20/10/2002</date-
inscription>
         <nbmsg>1721</nbmsg>
         <nbtutoriel>4</nbtutoriel>
     </activite>
   </membre>
   <membre>
      <nom/>
      <pseudo>jpdar</pseudo>
      <activite>
         <date-inscription>29/12/2005</date-
inscription>
         <nbmsg>14</nbmsg>
         <nbtutoriel>0</nbtutoriel>
      </activite>
   </membre>
   <membre fonction="admin">
      <nom>Cédric Chatelain</nom>
      <pseudo>cchatelain</pseudo>
      <activite>
         <date-inscription>26/05/2002</date-
inscription>
         <nbmsg>2840</nbmsg>
         <nbtutoriel>10</nbtutoriel>
      </activite>
   </membre>
</developpez>
```

1.2. Un peu de vocabulaire

Pour ceux qui n'auraient pas ou peu eu l'occasion de cotoyer le format XML, voici un survol très rapide du vocabulaire lié à cette norme. Pour plus de précisions, rendez-vous sur le forum XML : http://xml.developpez.com

Un document XML est reconnaissable par :

- Les éléments de balisage : un " élément " est composé d'une balise ouvrante, de la balise fermante associée et de tout ce qui se trouve entre ces balises.
- Une balise ouvrante peut aussi contenir un ou plusieurs " attributs "
- Les noeuds : essence même du document XML, ils modélisent tous les éléments et le document lui-même est modélisé par le noeud " / " appelé "racine du document XML"

En reprenant notre exemple précédent, nous pouvons donc dire :

```
<!---Balises ouvrantes/fermantes--->
<developpez>  </developpez>

<!---Attributs : fonction dans la balise membre--->
<membre fonction="Responsable Visual Basic">
</membre>

<!---Eléments--->
   <membre>
      <nom/>
      <pseudo>...</pseudo>
      <nbmsg>...</nbmsg>
   </membre>
```

Il ne sert à rien d'aller plus loin dans le détail des normes XML, vous trouverez tout ce qui vous intéresse sur le forum XML ainsi qu'en lisant les tutoriels XML de developpez.com : Lien39

1.3. Quelques bonnes raisons d'utiliser le XML

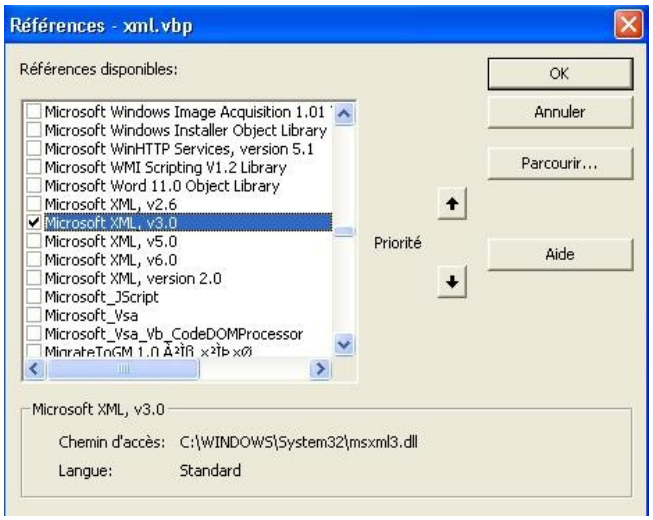
Il en existe beaucoup et son utilisation dépend exclusivement du type d'application mais, sachez que cette norme peut être invariablement utilisée pour les raisons suivantes :

- la facilité de lecture d'un document XML par qui que ce soit
- le nombre sans cesse croissant des applications qui utilisent et optimisent cette norme
- la simplicité d'utilisation sur Intranet/Extranet
- la rapidité de création d'une structure XML
- l'aisance d'exploitation via le code de ce type de document
- l'inutilité d'une formation pour encoder des données sous un tel format
- importation et exportation de données à partir et vers de multiples applications
- ...

2. Exploitation de document XML

2.1. Les références

Comme pour tout composant non référencé à la base dans Visual Basic 6, il faut évidemment cocher les bonnes références avant de pouvoir utiliser la norme XML. Au fil du temps, cette norme a poursuivi son évolution dans les systèmes d'exploitation, il existe donc plusieurs références. Dans ce petit projet, j'ai choisi Microsoft XML 3.0 car elle est installée automatiquement avec Internet Explorer 6 et vous n'aurez ainsi aucun problème pour exécuter ce programme.



Microsoft XML v2.6

2.2. Le principe d'évolution dans un document XML

Comme vous avez pu le constater sur l'image représentant l'arbre de notre document XML, le principe d'élaboration est basé sur des niveaux de balises. Bien visualiser la logique de cette cascade de noeuds est essentiel pour développer le code d'exploitation. En effet, le principe sera toujours le même quelque soit la manipulation à effectuer sur le document : il faudra progresser logiquement dans les noeuds, niveau par niveau. Vous ne pouvez en effet pas atteindre directement un noeud de niveau 3 sans avoir référencé les noeuds de niveau supérieur au préalable.

Les puristes de la norme XML n'aimeront peut-être pas ce genre de "nivellage" mais il est bien plus facile de s'y retrouver dans la programmation en ayant bien en vue le nombre de noeuds à définir pour progresser dans le document.
Par exemple :

```
Dim oDvpDOMDocument As MSXML2.IXMLDOMDocument
Dim oNoeudMembre As MSXML2.IXMLDOMElement
' élément membre
Dim oNoeudEnfantMembre As MSXML2.IXMLDOMElement
' élément fils de oNoeudMembre
Dim oNoeudEnfantMembreActivite As MSXML2.IXMLDOMElement
' élément fils de oNoeudEnfantMembre
```

Ceci est évidemment un exemple car, parfois, il n'est pas utile de déclarer tous les noeuds enfants, on peut passer d'un noeud de même niveau à un autre noeud de même niveau sans déclarer les noeuds enfants.

2.3. Les principales déclarations liées au XML

Cette liste est réduite à sa plus simple expression aussi bien pour les éléments existants que pour les méthodes et propriétés associées.

2.3.1. MSXML2.DOMDocument

Permet de déclarer une variable comme faisant référence au document XML utilisé.
Cette variable, reconnue comme étant un document XML, possède des méthodes qui permettent d'exploiter le document.
Par exemple, si je désire ajouter un élément à mon document, je pourrai le programmer comme suit :

```
Set oNoeudMembre =
oDvpDOMDocument.createElement("membre")
```

en spécifiant grâce au Set à quel noeud j'ajoute un élément.

2.3.2. MSXML2.IXMLDOMElement

Concerne bien évidemment les éléments du document XML et donc les "niveaux" que je peux parcourir. Voir de nouveau l'exemple de "nivellage".
Ces éléments possèdent eux aussi des méthodes permettant de réaliser diverses actions. Les plus simples à comprendre et à appréhender me semblent être :

- getAttribute : permet de lire l'attribut de l'élément
- getElementsByTagName : permet de rechercher la valeur d'un élément par le nom de la balise
- hasChildNodes : permet de savoir si des noeuds enfants existent et si oui, quels sont-ils.

2.3.3. MSXML2.IXMLDOMNode

Comme le nom l'indique, nous travaillons ici au niveau des noeuds de notre document XML qui possèdent eux aussi des méthodes comme :

- hasChildNodes : permet de savoir si ce noeud précis possède des noeuds enfants
- removeChild : permet de supprimer un noeud enfant
- replaceChild : permet de remplacer un noeud enfant par une autre valeur
- selectSingleNode : permet de sélectionner un noeud précis (grâce au nom de la balise)

2.3.4. MSXML2.IXMLDOMNodeList

La liste des noeuds, comme vous l'avez sûrement deviné, permet de faire des recherches assez rapides dans les noeuds enfants.
Un petit exemple pour illustrer ce fait :

```
Dim oListeNoeuds As MSXML2.IXMLDOMNodeList
Dim oNoeud As MSXML2.IXMLDOMNode

For Each oNoeud In oListeNoeuds
 ......
Next
```

Bien évidemment, il existe aussi des propriétés comme dans l'exemple ci-dessus où la valeur d'un noeud est définie par la propriété "Text".

2.4. Conclusion

Ce petit tour d'horizon est bien trop rapide pour vous faire plonger dans toutes les possibilités des méthodes et propriétés de l'objet XML.
Pour vous familiariser avec leur utilisation, je vous propose quelques petits codes tout simples, juste pour bien comprendre la logique d'exploitation d'un document XML.

3. Quelques codes simples

3.1. Ouvrir un document XML

Il faut juste attirer l'attention sur le fait qu'il existe deux manières de charger un document XML : synchrone ou asynchrone.
Par défaut, le fichier est chargé de manière asynchrone. Il faut spécifier :

```
oDvpDOMDocument.async = False
```

pour charger tout le document en mémoire. Ce sera le cas dans les exemples associés au tutoriel vu que le fichier est en local et très petit.

Le document sera chargé pour chaque action choisie dans l'exemple afin de ne pas vous obliger un ordre d'exécution précis des exemples fournis.
Vous pouvez donc choisir n'importe quelle action de l'application en exemple pour voir le code d'ouverture et de chargement d'un document XML.

3.2. Rechercher une donnée

Le principe de la recherche consiste à parcourir les noeuds à partir de la racine, soit en cherchant une balise précise, soit en cherchant un attribut.

Il faut donc bien veiller à définir les "elements", "nodes", "nodelist" dont on peut avoir besoin. Evidemment, dans l'exemple c'est assez simple puisque l'on connait la structure du document et le nom des balises.
La programmation est moins aisée s'il faut partir d'un document XML dont on ne connait pas la structure. Ce n'est pas le but dans cette introduction.
L'exemple joint recherche un pseudo précis.

3.3. Ajouter une donnée

Dans notre exemple, cela consiste à ajouter un nouveau membre dans le XML des membres de developpez.com.
L'ajout concerne se fait toujours en signalant à la racine du document XML que l'on va créer un nouvel élément.
Dès que vous avez compris cela, trouver le code est relativement simple sachant que la procédure est la même pour chaque niveau de création d'une balise.

Il faut bien comprendre que pour créer un nouvel élément, il faut 2 instructions. La méthode "createElement" ne crée pas la balise, elle n'est vraiment implémentée qu'au moment de l'exécution de la méthode "appendChild".

3.4. Mise à jour d'une donnée

La mise à jour d'une donnée reste basée sur le même principe que la recherche à la seule différence que l'on modifie la donnée contenue dans la balise ciblée. On procède alors en modifiant la propriété Text, ce qui est une chose assez aisée à réaliser.

3.5. Créer un document XML

Très similaire à l'ajout de données sauf qu'il faut créer le document en y plaçant l'entête propre à ce type de document ainsi que l'extension ad hoc.

3.6. Gestion des erreurs

Un simple exemple de la gestion des erreurs via "parseError".

4. Tutoriel interactif

Je vous propose, sur base de l'exemple associé à ce tutoriel, de le rendre interactif.
L'exploitation de documents XML étant un domaine très vaste, j'ai pensé que le mieux pour aider les utilisateurs serait de pouvoir faire évoluer ce tutoriel avec vos propres réalisations.
Pour ce faire, il faut évidemment instaurer quelques règles d'organisation et préciser comment vous allez pouvoir étoffer ces codes plutôt simplistes.

Il est nécessaire de poser les bases de l'organisation afin que ce système soit performant et intéressant à tous points de vue.
Si vous désirez que votre proposition soit publiée, je vous enjoins donc à suivre ces quelques impératifs :

4.1. Comment organiser vos propositions ?

En quelques points, voici comment procéder :

- Se baser sur l'exemple joint à ce tutoriel en y ajoutant une feuille portant votre nom/pseudo.
- Se baser sur le document XML utilisé. Vous pouvez le modifier suivant vos besoins mais, dans ce cas, faites-en une copie et appellez-le "dvp-xxxx.xml" ou "xxx" est votre nom/pseudo
- Lancer l'exécution de votre feuille à partir de la feuille Sélection en y ayant placé un nouveau bouton avec un titre explicite
- Commentez clairement votre code sans user d'abréviations.

4.2. Comment rédiger vos explications ?

Il est bien évident que les codes seuls ne peuvent pas renseigner sur ce que vous avez voulu réaliser. Une explication substantielle est donc nécessaire en accompagnement.
Veuillez respecter ces quelques consignes dans vos explications :

- Précisez sur quel système d'exploitation vous travaillez ainsi que tout autre renseignement qui pourrait affecter l'exécution du code.
- Décrivez précisemment quelles références sont nécessaires et prenez la peine éventuellement de faire une capture d'écran pour montrer la(es) référence(s) cochées.
- En début de vos explications, faites un résumé succint du but de votre code.
- Soyez clair et précis dans le texte explicatif, évitez les abréviations et le style sms.
- Tapez le texte dans un traitement de texte d'utilisation courante voire même dans le bloc-note, cela est largement suffisant. La mise en page n'est pas nécessaire (sauf si vous désirez chapitrer le texte), elle sera de toute façon adaptée au format actuel du tutoriel.

4.3. Comment me faire parvenir votre contribution ?

Afin d'intégrer votre proposition à ce tutoriel, il est nécessaire que je recoive, sous forme de fichiers "ZIP" :

- Le programme d'exemple initial complété par votre(vos) feuille(s) et zippé dans un fichier nommé "xml.zip".
- Votre fichier d'explications zippé dans un fichier nommé "expl-xxx.zip" où "xxx" est votre nom/pseudo.
- Eventuellement un fichier zippé contenant vos images sous le nom "images-xxx.zip" où xxx est votre nom/pseudo.
- Prévenez-moi par MP pour me signaler votre participation. Envoyez-moi les zip sur mon mail rédaction ou donnez-moi un lien internet où les télécharger. (Vous pouvez regrouper les différents *.zip

dans un seul zip portant votre nom/pseudo).
- Toute communication concernant votre proposition avant ou après réalisation se fera exclusivement par MP.

4.4. Intégration au tutoriel

Je me charge de l'intégration de vos commentaires et du programme modifié dans ce tutoriel en référencant, bien entendu, que vous en êtes l'auteur exclusif. Les sources publiées sont toujours libres de droit mais le texte associé est protégé au même titre que tous les tutoriels publiés sur développez.

Je précise aussi que l'intégration de votre code est soumis à l'approbation de l'équipe Visual Basic quant à sa nécessité, la manière de coder, la pertinence des commentaires et explications ainsi que la facilité d'exécution. Il est évident que nous vous tiendrons au courant des remarques éventuelles afin de faire évoluer positivement votre proposition et aboutir à son intégration.

Quelques précisions importantes :

- Votre code doit répondre à une utilisation générale assez courante et être complet quant à la solution proposée.
- Ne seront acceptés que les codes proprement indentés et commentés.
- Ne seront acceptés que les codes accompagnés d'un fichier d'explications claires et complètes.
- Le code doit être testé et fonctionnel sous une installation standard de Visual Basic 6.0.
- L'acceptation ou le refus d'un code est du ressort de l'ensemble de l'équipe Visual Basic. Il ne sera pris en compte aucune demande de renseignements, réclamation, question ou quelconque correspondance qui ne serait pas rédigée suivant les règles de developpez.com (sms, orthographe, politesse, ...).

A vos codes et bon travail !

Retrouvez les téléchargements, le code source des exemples et l'article en ligne de Cécile Muno :
Lien40

Vu dans la FAQ

Comment supprimmer les doublons dans un affichage ?

On utilise la "Méthode Muench" : création d'une clé indexée par la valeur du noeud, et utilisation de la fonction generate-id() pour différencier les noeuds afin de ne sélectionner que la première occurrence d'un noeud parmi ceux ayant la même valeur.

Exemple avec un tri sur le contenu d'une balise Id

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:key name="id" match="Id" use="."/>
<xsl:template match="/">

    <xsl:for-each select="//Id[generate-id(.)=generate-
       id(key('id', .)[1])]">
      <xsl:value-of select='.'/>
   </xsl:for-each>

</xsl:template>
</xsl:stylesheet>
```

Exemple avec un tri sur le contenu de l'attribut val d'une balise Id

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:key name="id" match="Id" use="@val"/>
<xsl:template match="/">

    <xsl:for-each select="//Id[generate-id(.)=generate-
       id(key('id', @val)[1])]">
      <xsl:value-of select='.'/>
   </xsl:for-each>

</xsl:template>
</xsl:stylesheet>
```

Pourquoi la fonction name(ma_selection) ne me renvoie par les résultats pour tous les noeuds séléctionnés ?

La fonction name(), mais aussi local-name() et namespace-uri () ne renvoie la valeur que du premier noeud sélectionné si plusieurs lui sont fournis en paramètre

Comment regrouper les noeuds en fonction de leurs éléments qui se ressemblent ?

il suffit d'opérer une variante de l'utilisation de la méthode Muench : http://xml.developpez.com/faq/?page=3#xslt_doublon

le xml

```
<?xml version="1.0" encoding="UTF-8"?>
<r>
	<t>
		<a>1</a>
		<b>test a</b>
	</t>
	<t>
		<a>2</a>
		<b>test b</b>
	</t>
	<t>
		<a>1</a>
		<b>test c</b>
	</t>
	<t>
		<a>2</a>
		<b>test d</b>
	</t>
</r>
```

le xslt

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
	<xsl:output method="xml" version="1.0"
encoding="UTF-8" indent="yes"/>
	<xsl:key name="regrouper" match="a" use="."/>
```

```
	<xsl:template match="/">
		<r>
			<xsl:apply-templates
select="r/t/a[generate-id(.)=generate-
id(key('regrouper',.)[1])]"/>
		</r>
	</xsl:template>
	<xsl:template match="a">
		<t>
			<xsl:copy-of select="."/>
			<xsl:apply-templates
select="//b[../a=current()]"/>
		</t>
	</xsl:template>
	<xsl:template match="b">
		<xsl:copy-of select="."/>
	</xsl:template>
</xsl:stylesheet>
```

le résultat

```
<?xml version="1.0" encoding="UTF-8"?>
<r>
	<t>
		<a>1</a>
		<b>test a</b>
		<b>test c</b>
	</t>
	<t>
		<a>2</a>
		<b>test b</b>
		<b>test d</b>
	</t>
</r>
```

Comment tester si une valeur est NaN (No a Number) ?

Soit $n une variable de type quelconque :

```
<xsl:choose>
  <xsl:when test="string(number($n)) = 'NaN'">$n n'est
pas un nombre !</xsl:when>
  <xsl:otherwise>$n est un nombre</xsl:otherwise>
</xsl:choose>
```

Est-il possible de spécifier des entités dans un XML Schema ?

Ce n'est pas exactement possible comme on le faisait pour un DTD. Cependant, il existe deux solutions (recommandée par le W3C) qui permet de simuler ce comportement :

La première consiste a insérer un DTD contenant la dite entité au début du XML :

```
<?xml version="1.0" ?>
<!DOCTYPE root [
<!ENTITY eacute "&#xE9;">
<!ENTITY egrave "&#xE8;">
]>
<root xmlns="http://www.exemple.com/XMLNameSpace">
	<!-- etc... -->
		<town>Moll&eacute;g&eagrave;s</town>
	<!-- etc... -->
</root>
```

Bien sûr, ceci est un peu pénible car il faut le spécifier au début de chaque document XML.

D'où la deuxième solution, elle permet de simuler une entité en déclarant une balise à contenu fixe approprié :

```
<xsd:element name="eacute" type="xsd:token"
fixed="&#xE9;"/>
```

Pour l'utiliser ensuite dans le document XML, il suffit de le faire ainsi (pour des raisons de lisibilité, nous avons préférer retrancher les pseudo-entitées dans un XSD séparé :

```
<?xml version="1.0" ?>
<root xmlns="http://www.exemple.com/XMLNameSpace"
xmlns:E="http://www.exemple.com/Entites>
	<!-- etc... -->

	<town>Moll<E:eacute/>g<E:eagrave/>s</town>
	<!-- etc... -->
</root>
```

Est-il possible de faire dépendre le nombre d'éléments [éventuellement d'un élément en particulier] dans les fils d'un élément ?

NON...
Il est impossible de faire quelque chose qui forcerait une structure dans ce style :

```
<a b="1" c="1">
	<b>a</b>
	<c>a</c>
</a>
<a b="2" c="1">
	<b>a</b>
	<b>a</b>
	<c>a</c>
</a>
<a b="1" >
	<b>a</b>
</a>
```

Retrouvez la FAQ XML : Lien42

Changement des mots de passe système pour Oracle

Cet article va vous montrer comment changer proprement et facilement un mot de passe d'un compte système Oracle

1. Principe de base

Modifier un mot de passe chez Oracle ? Rien de plus simple... Il s'agit cependant de ne pas oublier l'impact que cela peut avoir sur les modules Oracle qui les utilisent et qui les stockent (bien entendu de façon cryptée) dans des fichiers.

Se connecter en tant que sys as sysdba

```
alter user sys identified by NouveauMotDePasse ;
alter user system identified by NouveauMotDePasse ;
alter user dbsnmp identified by NouveauMotDePasse ;
alter user sysman identified by NouveauMotDePasse ;
```

Pour dbsnmp, ne pas utiliser 8 caractères pour le mot de passe, ni des caractères autres que alpha-numériques compte tenu d'un bug répertorié.

2. Streams

Cas particuliers pour la réplication si vous utilisez Streams, et plus particulièrement Downstreams (capture et apply du côté cible).

Dans ce cas précis, le mot de passe de SYS doit être obligatoirement strictement identique sur le serveur source et le serveur cible.

3. Utilisateur dbsnmp

3.a Agent 9i

Dans

```
snmp.connect.{Nom global}.name=dbsnmp
snmp.connect.{Nom global}.password=NouveauMotDePasse
```

... puis redémarrer l'agent

Source : Lien42

3.b Agent 10g

Connecté en tant qu'admin et en mode console, stopper la console via

```
ecmt1 stop dbconsole
```

Modifier le mot de passe en tant que sysdba

```
alter user dbsnmp identified by NouveauMotDePasse
```

Editer le fichier *${ORACLE_HOME}/`hostname`_${ORACLE_SID}/sysman/emd/targets.xml* et modifier les lignes suivantes:

```
<Property NAME="UserName" VALUE="2ed7f792e30adc89"
ENCRYPTED="TRUE"/>
<Property NAME="UserName" VALUE="c8d4082a472b36ae"
ENCRYPTED="TRUE"/>
```

en

```
<Property NAME="UserName" VALUE="dbsnmp"
ENCRYPTED="TRUE"/>
<Property NAME="UserName" VALUE="NouveauMotDePasse"
ENCRYPTED="TRUE"/>
```

A noter que dans le cas d'une installation RAC, vous aurez chaque fois ces deux entrées à double. Une fois sous la rubrique

```
<Target TYPE="oracle_database" NAME="Votre Global
Name">
```

... et une fois sous la rubrique

```
<Target TYPE="rac_database" NAME="Votre Global Name">
```

Redémarrer la console : ce redémarrage aura pour effet de crypter les valeurs précédemment en clair et de passer le flag ENCRYPTED à TRUE.

```
ecmt1 start dbconsole
```

Pour déterminer si les nouveau paramètes ont bien été pris en considération lors du redémarrage, et pour des raisons évidentes de sécurité, allez à nouveau éditer le fichier *${ORACLE_HOME}/`hostname`_${ORACLE_SID}/sysman/emd/targets.xml* pour vérifier que tout mot de passe soit bien crypté.

Source : Lien43

4. Utilisateur sysman

4.a Agent 10gR1

Connecté en tant qu'admin et en mode console, stopper OMS via

Modifier le mot de passe en tant que sysdba

```
alter user sysman identified by NouveauMotDePasse
```

Editer le fichier *${ORACLE_HOME}/sysman/config/emoms.properties* et modifier les lignes suivantes:

```
oracle.sysman.eml.mntr.emdRepPwd=NouveauMotDePasse
oracle.sysman.eml.mntr.emdRepPwdEncryption=FALSE
```

Redémarrer OMS

```
ecmt1 start oms
```

S'assurer que *oracle.sysman.eml.mntr.emdRepPwd* est maintenant encrypté.

Source : Lien44

4.b Agent 10gR2

Connecté en tant qu'Oracle sur la console console, stopper La console via

```
ecmt1 stop dbconsole
```

Dans le fichier ${ORACLE_HOME} /`hostname`_${ORACLE_SID}/sysman/config/emoms.properties, modifier les lignes en conséquent. Remplacez

```
oracle.sysman.eml.mntr.emdRepUser=SYSMAN
oracle.sysman.eml.mntr.emdRepPwd=d0355495a68cd5ae
oracle.sysman.eml.mntr.emdRepPwdEncryption=TRUE
```

par

```
oracle.sysman.eml.mntr.emdRepUser=SYSMAN
oracle.sysman.eml.mntr.emdRepPwd=NouveauMotDePasse
oracle.sysman.eml.mntr.emdRepPwdEncryption=FALSE
```

Redémarrer la console via

```
ecmt1 start oms
```

En rééditant le fichier ${ORACLE_HOME} /`hostname`_${ORACLE_SID}/sysman/config/emoms.properties, s'assurer que le mot de passe sous oracle.sysman.eml.mntr.emdRepPwd soit crypté et que le flag oracle.sysman.eml.mntr.emdRepPwdEncrypted soit sur TRUE.

5. Résolution de problèmes connus

Si vous avez effectué vos changements de mot de passe avant l'arrêt de la console, celle-ci peut avoir passé vos comptes - LOCKED(TIMED). Assurez-vous donc que les comptes SYSMAN et DBSNMP soient bien déverrouillés lorsque vous lancez votre console.

Retrouvez l'article de Fabien Celaia en ligne : Lien45

Gestion des transactions avec SQL Server

Cet article vous apprendra à garantier/préserver l'intégrité de vos données lors de l'exécution d'un ensemble de requêtes sur votre base de données.

1. Introduction

Vous avez souvent du effectuer une série de requêtes sur une base de données et vous connaissez les risques que vous encourez si l'une d'entre elles échoue et qu'une autre réussit. Vous risquez d'avoir des données incohérentes et de perdre du temps ensuite pour trouver d'ou vient le problème.

Cet article va vous donnez une manière de faire très simple pour sécuriser vos requêtes SQL-Server.

2. Sécurisation

2.1 Transaction

Une transaction est un ensemble de requêtes que l'on regroupe en une seule unité logique de travail qui pourra ensuite être, soit validée, soit annulée.

La première chose à faire dans ce cas-là est donc d'encapsuler vos requ êtes dans une transaction. On va donc faire démarrer une transaction au début et la committer à la fin :

```
BEGIN TRANSACTION changement_etat_civil
-- Vos requêtes
COMMIT TRANSACTION changement_etat_civil
```

Mais cela ne change rien au problème, en cas d'erreur, l'ensemble est tout de même committé et nous avons toujours le risque d'avoir des données incohérentes.

2.2 Détection des erreurs

Nous allons donc ajouter une partie de gestion d'erreurs à notre code. Pour récupérer une erreur, il suffit d'employer la variable prédéfini @@ERROR qui contient l'erreur pour la dernière requête effectuée. Nous allons donc déclarer une variable @errors pour stocker les différentes erreurs récupérées lors de l'exécution :

```
DECLARE @errors INT

UPDATE T_TEXTES SET TEXTE = 'Divorcé(e)' WHERE ID = 1
SET @errors = @errors + @@error
```

2.3 Gestion des erreurs

Bon, c'est bien beau, vous me direz, on voit les erreurs, mais en cas d'erreurs notre base est toujours incohérente. Bien, ca prouve que vous suivez. Une astuce pour cela est de déclarer notre variable @errors à 1, comme ça, si à la fin de l'exécution des requêtes, elle n'est plus à un, on sait qu'il y a eu une erreur et on peut faire quelque chose.

```
DECLARE @errors INT
SET @errors = 0
```

On va maintenant contrôler s'il y a eu ou non des erreurs et s'il y en a eu, on va annuler toutes les opérations faites dans la transaction en utilisant la commande ROLLBACK TRANSACTION :

```
IF @errors = 0
        COMMIT TRANSACTION changement_etat_civil
ELSE
        ROLLBACK TRANSACTION changement_etat_civil
```

2.4 Points d'enregistrements

Une autre chose qu'il est utile de connaître est le fait qu'on peut insérer des savepoint dans une transaction, c'est à dire un point d'enregistrement, sur lequel on peut se baser pour faire un RollBack. Ainsi, on n'a pas besoin de faire une annulation sur l'intégralité de la transaction, si on a deux parties bien distinctes dans notre script, on peut intercaler un point d'enregistrement entre les 2. Si la première partie s'est déroulée sans problèmes, mais que la deuxième partie s'est mal passé, on faire un rollback jusqu'à notre point d'enregistrement puis un commit. Ainsi, tout ce qui s'est bien passé est validé et le reste est annulé.

Pour sauvegarder un point d'enregistrement, il vous suffit de faire ceci

```
SAVE TRANSACTION nom_du_savepoint
```

Et si ensuite, vous voulez revenir à cet état, il vous suffit de faire :

```
SAVE TRANSACTION nom_du_savepoint
```

3. Conclusion

En conclusion, il n'est vraiment pas difficile de sécuriser du code SQL-Server, il suffit d'un peu de rigueur et vous aurez ainsi supprimé une bonne part de surprise et aurez des codes plus sûrs à exécutez.

Retrouvez l'article de Baptiste Wicht en entier : Lien46

Livres

SQL De Frédéric Brouard et Christian Soutou

Critique par Damien Griessinger

Efficacité et simplicité, tels sont les maîtres mots de cet ouvrage réalisé par deux références de l'informatique française : Frédéric BROUARD et Christan SOUTOU.

Ce livre sort des sentiers battus et propose à tous lecteurs un référentiel sur le SQL en général et un très bon support de cours. Clairement tourné vers les étudiants, on retrouve de nombreux schémas et exemples, chaque thème abordé finissant par des exercices corrigés.

Les professionnels ne sont pas en reste, les auteurs n'hésitent pas à démontrer la théorie par des requêtes, ce qui est un énorme gain de temps.

Le format du livre est certes non conventionnel car il fait presque du A4 mais le choix est judicieux, la mise en page très bien faite permet une facilité de lecture que l'on retrouve rarement chez les concurrents.

J'ai rarement vu d'ouvrage traitant sur le SQL aussi complet, clair et avec un aussi bon rapport/qualité prix, je ne saurais que le recommander de toute urgence !

Retrouvez tous les livres SGBD : Lien47

Les derniers tutoriels et articles

Création d'un gadget pour la Windows Sidebar

Article présentant les Gadgets de la Windows Sidebar de Windows Vista, ainsi que l'explication de leur développement.

1. Introduction

Depuis maintenant quelques années, et avouons-le copiés sur Mac OS, les compléments de bureau se développent dans tous les sens. On peut voir des docks, des menus complémentaires, ou encore des Widgets, sortes de petits éléments à fonctions bien précises que l'on place où l'on veut sur le bureau.
Force est de constater que ce genre de produits complémentaires sont utiles et très demandés par les utilisateurs, Microsoft a décidé d'intégrer les Widgets à son futur système d'exploitation: Windows Vista.

Microsoft a apporté sa petite touche à l'édifice en développant la Windows Sidebar, une barre se trouvant sur le côté de l'écran, accrochée au bureau. C'est sur cette Sidebar que peuvent déposés les Widgets nommés pour l'occasion: les Gadgets. Ces gadgets peuvent être placés aussi bien sur la Sidebar dans une forme réduite (nous l'appellerons dockée), que sur le bureau dans leur version complète (non dockée).
Voici ce à quoi peut ressembler votre Sidebar:

Bien que différents gadgets déjà tout faits puissent être trouvés sur Internet, nous allons nous intéresser à la création de notre propre gadget.

2. Principe du gadget

Qu'est qu'un gadget? Quelle technologie cela utilise? Quels outils nous faut-il pour en développer un? Toutes ces questions qui, je suis sûr, trottent dans votre tête, et auxquelles je vais tâcher de répondre.
Tout d'abord, un Gadget est un fichier ayant pour extension .gadget. Il s'agit d'un fichier compressé ZIP dont l'extension a été changée. Il contient différents éléments qui, suivant des règles strictes, permettent de former un Gadget fonctionnel.

Niveau technologie, le Gadget utilise l'HTML, les css, le JavaScript mais peut également utiliser du Flash ou de l'ActiveX.

Quant aux outils, pour réaliser cet article, je vous demanderai de vous munir d'une souris, d'un clavier et de n'importe quel éditeur de texte: Notepad faisant très largement l'affaire. Oui, il ne faut rien de plus.

Passons à la structure du Gadget. Celui-ci est constitué d'un certain nombre d'éléments dont certains sont obligatoires et d'autres non. Voici les éléments possibles:

- Le Manifest (gadget.xml) : Contient les informations sur le Gadget
- Fichier html (XXX.html) : Contient le code principal du Gadget servant principalement à définir l'affichage du Gadget
- Fichier de paramètres (Settings.html) : Permet de contenir les paramètres de l'utilisateur (si votre gadget a des options)
- Fichiers images (XXX.png) : Icone représentant le Gadget dans la boîte à gadgets
- Fichiers ressources (*.html, *.js, ...) : Différentes ressources pouvant être utilisées par le Gadget

3. Mon premier Gadget

3.1 Explication

Avant de créer un vrai Gadget, nous allons créer un Gadget tout ce qu'il y a de plus simple et qui affichera un simple message. Lors du développement de ce Gadget, nous verrons un à un les différents éléments noyaux d'un Gadget. Dans un second chapitre, nous ferons évoluer ce Gadget pour le rendre plus interactif.

Passons aux choses sérieuses...

3.2. Développement

Créez n'importe où un dossier que vous nommerez comme bon vous semble.
Entrez alors dans ce dossier et créez-y un nouveau fichier texte que vous nommerez gadget.xml, puis ouvrez ce fichier avec Notepad. Collez-y tel quel le code suivant

gadget.xml

```
<?xml version="1.0" encoding="utf-8" ?>
    <gadget>
        <name>Mon premier gadget</name>
        <namespace>Developpez</namespace>
        <version>1.0</version>
        <author name="pharaonix">
             <info url="www.developpez.com" />
             <logo src="logo.png"/>
        </author>
        <copyright>2006</copyright>
        <description>C'est mon premier gadget qu'il est
tout beau tout frais ;o)</description>
            <icons>
```

```
            <icon height="48" width="48"
src="pharaonix.png"/>
        </icons>
        <hosts>
            <host name="sidebar">
                <base type="HTML" apiVersion="1.0.0"
src="dvp.html" />
                <permissions>Full</permissions>
                <platform minPlatformVersion="0.3" />
            </host>
        </hosts>
        <version value="1.0.0.0"
MinPlatformVersion="0.1"/>
    </gadget>
```

Expliquons-le maintenant. Nous remarquons tout d'abord la balise racine <gadget> dans laquelle se trouveront tous les éléments de notre Gadget.
Ainsi nous avons tout d'abord la balise <name> qui contient le nom de l'auteur
La balise <namespace> qui contient généralement le nom de l'entreprise. Balise utilisée dans le cas où vous créez plusieurs gadgets
La balise <version> qui contient le numéro de version du gadget
La balise <author> qui contient les informations sur l'auteur. Son nom (attribut name), son site web (balise <info>, attribut url) et son logo (balise <logo>, attribut src)
La balise <copyright> qui contient l'année du copyright
La balise <description> qui comme son nom l'indique contient une phrase décrivant les fonctions de votre gadget.
Nous avons ensuite la balise <icones> qui nous sert à définir l'icône qui représentera notre Gadget dans la boîte à gadgets.
La dernière partie est configurée pour signaler que ce sera un gadget Windows (et non Web). La seule chose chose à remarquer est l'attribut src de la balise base qui contient dvp.html

Toujours dans le dossier précédemment créé, créez maintenant un fichier que vous nommerez dvp.html. Ouvrez-le avec Notepad et collez-y le code suivant:

dvp.html
```
<html>
<style>
  body
  {
    width:130;
    height:50;
  }
</style>
<body>
 Vive Developpez.com !
</body>
</html>
```

Vous avez remarqué que dans le fichier XML, nous citions deux images. La première est obligatoire; je l'ai nommée pharaonix.png. Ajoutez donc un fichier png nommé ainsi. La deuxième image est facultative mais il peut parfois être intéressant d'y mettre son logo personnel ou le logo de son entreprise. Ajoutez aussi une image PNG de votre choix, que vous nommerez logo.png.
Votre dossier devra alors contenir les éléments suivants :

- pharaonix.png
- logo.png
- gadget.xml
- dvp.html

Sélectionnez maintenant les fichiers et créez une archive ZIP que vous nommerez dvp.gadget (sachant que "dvp" peut être le nom que vous voulez). Vous obtenez votre premier gadget et c'est sous cette forme que vous devrez le distribuer si vous voulez en faire profiter d'autres personnes.

3.3. Installation

Il faut maintenant tester notre gadget. Cette étape demande beaucoup de doigté, puisqu'il suffit de double-cliquer sur le gadget pour qu'une boîte de dialogue s'ouvre, et demande la confirmation d'installation du gadget.
Cliquez donc sur Installer.

Le fichier zip est alors dézippé dans un dossier qui sera appelé VotreNamespace.Nomdugadget.gagdet. Dans mon cas: developpez.dvp.gadget.
Ce dossier se trouve ici: H:\Users\%username%\Local Settings\Microsoft\Windows Sidebar\Gadgets

C'est dans ce dossier que sont placés les gadgets installés et qui pourront être choisis dans la boîte à gadgets de la Windows Sidebar.
Ouvrez maintenant la Sidebar et ouvrez la boîte d'ajout de gadgets, vous devriez voir votre gadget dans la liste. Cliquez sur détails et vous retrouverez les informations sur le gadget et l'auteur que vous aviez saisi dans le fichier XML.

Faites le glisser jusqu'à votre Sidebar et si tout va bien, votre premier gadget devrait apparaitre tel quel:

Et voilà, vous venez de créer votre premier Gadget! Essayons maintenant d'aller plus loin.

4. Création avancée

Nous allons ici créer un gadget qui ne vous sera pas forcément très utile, mais qui vous permettra de voir les différentes fonctionnalités que l'on peut donner à gadget. Ainsi, nous allons tenter de créer une barre de recherche pouvant lancer des recherches sur le site de son choix.
Cela aura pour effet de nous faire voir différents principes, comprenant l'interaction avec Internet, le design du Gadget et l'utilisation d'options sur un Gadget.

4.1. Les options

Reprenez le code HTML et rajoutez-lui une balise <head> dans laquelle nous déclarerons un fichier CSS et un fichier JavaScript. Notez que nous avons retiré la balise <style> car nous allons la placer dans le fichier CSS.

dvp.html
```
<html>
```

```
<head>
  <link href="dvp.css" rel="stylesheet" type="text/css"
/> <!--  nom de notre css (que l'on créera juste après
-->
  <script src="dvp.js" language="JavaScript"></script>
<!--  nom de notre fichier javascript (que l'on créera
juste après -->
</head>
<body>
 Vive Developpez.com !
</body>
</html>
```

Créez dans le répertoire un fichier dvp.css et éditez-le avec Notepad et collez-y le code suivant:

dvp.css

```
body {
    width:130;
    height:50;
}
```

Nous compléterons la feuille CSS dans le chapitre Design. Créez maintenant un fichier texte vide que vous nommerez settings.html. Puis ouvrez le fichier dvp.html pour y coller dans la balise body le code suivant:

dvp.html

```
<script>
  System.Gadget.settingsUI = "settings.html";
</script>
```

Automatiquement le gadget détectera que des options sont possibles et le bouton option (la petite "clé à molette") s'activera.

Ouvrez le fichier settings.html précédemment créé et tapez-y ceci. Nous définissons l'affichage du panneau options, dans lequel nous plaçons une Textbox qui servira à stocker l'url du flux RSS. Nous nommerons cette Textbox txtUrl.

settings.html

```
<html>
<head>
 <style>
   body {
        width:160;
        height:50;
   }
 </style>
</head>
<body >
       Url:<br/>
        <input type="text" id="txtURL" length="100">
</body>
</html>
```

Dorénavant, lorsque vous cliquerez sur le bouton Options de votre Gadget, une fenêtre s'ouvrira, proposant de saisir l'url. Notre gadget devrait maintenant ressembler à ceci :

Nous allons coder trois fonctions en JavaScript, qui nous permettront de charger l'URL actuellement utilisée et de sauvegarder l'URL saisie par l'utilisateur. Commençons par la méthode init():

```
function init() {
    var temp = System.Gadget.Settings.read("vURL");
        if (temp  != "")
        txtURL.innerText = temp ;
    else txtURL.innerText = "http://www.developpez.com"
}
```

La méthode qui sauvegardera l'URL saisie

```
function Save() {
    System.Gadget.Settings.write("vURL", txtURL.value);
}
```

Et la méthode de fermeture qui sauvegardera lorsque la fenêtre des options se fermera.

```
System.Gadget.onSettingsClosing = SettingsClosing;
function SettingsClosing(event)
{
    if (event.closeAction == event.Action.commit) {
        Save();
    }
    event.cancel = false;
}
```

Revoyons le code que nous avons tapé. La méthode init() tente de récupérer la valeur de la variable vURL qui est sauvegardée par le gadget. Ne l'ayant pas encore défini, elle est vide et cela remplira avec la valeur par défaut "http://www.developpez.com".
Nous créons alors la méthode SettingsClosing() que nous "accrochons" à l'événement de fermeture de la fenêtre. Ainsi, cela appellera cette méthode uniquement pendant la fermeture. Cette méthode appelle alors la méthode save() qui sauvegarde le contenu de la Textbox dans la variable vURL, stockée par le gadget. Cette variable sera stockée par la Windows Sidebar dans le fichier Settings.ini qui se trouve dans le dossier **Local Settings\Microsoft\Windows Sidebar**. Ainsi, au prochain lancement, cette variable sera automatiquement chargée avec la dernière valeur enregistrée.

4.2. Le design

Nous allons remodifier le code du fichier dvp.html pour avoir un Gadget un peu plus esthétique. Récupérons d'abord sur Internet, une quelconque image de loupe, puis enregistrons-la dans le dossier de notre gadget. Nous allons alors ajouter l'image au code HTML, puis une Textbox et enfin un bouton:

```
<html>
<head>
  <link href="css/dvp.css" rel="stylesheet"
type="text/css" /> <!--  nom de notre css (que l'on
créera juste après -->
  <script src="js/dvp.js"
language="JavaScript"></script> <!--  nom de notre
fichier javascript (que l'on créera juste après -->
</head>
<script>
  System.Gadget.settingsUI = "settings.html";
</script>
<body>
<img src="/images/recherche.gif" /><span style="color:
#6389D8;font-weight: bold;font-size: 10pt;"> Rechercher
:</span>
<input id="txtSearch" type="text" size="14"
maxlength="30" />
<div align="right"><input type="submit" value="OK"
id="search" name="search" onclick="search()" /></div>
</body>
</html>
```

Notre gadget ressemblera dorénavant à ceci:

4.3 Le code

Niveau code, nous en avons déjà écrit pour les options mais il faut maintenant en écrire pour l'interface principale du gadget. Mais comme vous allez le voir, le plus dur a été fait. Nous allons simplement créer une méthode qui ouvre une fenêtre avec pour adresse, l'url des options concaténée avec la valeur de la Textbox de recherche, comme suit (placez le dans un fichier dvp.js):

dvp.js

```
function search(){
    var temp = System.Gadget.Settings.read("vURL");
    window.open(temp+txtSearch.value,'window');
}
```

Il nous reste plus qu'à appeler cette méthode. Pour cela, ajoutez la propriété onclick au bouton OK de dvp.html

dvp.html

```
<input type="submit" value="OK" id="search"
name="search" onclick="search()" />
```

A ce moment précis, votre gadget est fonctionnel et ira lancer une recherche sur le site web que vous avez passé en paramètre. Il faut bien entendu une URL qui accepte des mots en paramètre dans l'url (paramètres GET).
Il nous reste à fignoler notre gadget pour qu'il soit le plus réussi possible.

4.4 Finition

Voilà, nous avons un gadget qui marche mais avouons-le, il est pas très bien rangé. Nous (vous :)) allons donc créer un dossier css dans lequel nous déplacerons le fichier dvp.css, un dossier JavaScript où déplacer le fichier dvp.js, et un dossier images où placer toutes nos images. Une fois le rangement fait, il faudra éditer les chemins des fichiers, dans chaque fichier (principalement la css et dvp.html).

Sélectionnez tous les fichiers créés et comme précédemment, créez une archive ZIP que vous nommerez search.gadget. Double-cliquez dessus pour l'installer.
Maintenant, allez dans les options de votre gadget. Comme aucune URL n'a été précisée, la case se remplit automatiquement avec l'URL du dictionnaire de Developpez.com. Fermez alors les options et tapez un mot à rechercher. Une fenêtre du dictionnaire devrait s'ouvrir, lançant automatiquement la recherche avec le mot saisi. Et voilà, c'est fini :)

5. Création encore plus avancée

Dans cette dernière partie, nous allons voir deux derniers points qui se révéleront essentiels pour le développement d'un Gadget de qualité.

5.1. La localisation

En informatique, le terme localisation se rapporte à la langue de l'utilisateur. Ainsi, nous allons créer un gadget qui s'adaptera tout seul à la langue de l'utilisateur.

Nous avons ici un problème, c'est que nous affichons le mot "Rechercher". Mais que se passerait-il si le système sur lequel il était installé était en anglais? Notre utilisateur serait alors bien perdu. Nous allons faire en sorte que notre Gadget affiche un texte différent selon que l'ordinateur soit en français ou en anglais.

Chaque culture (langue) est représentée par deux parties, la langue et le pays. Pour la France, nous avons fr-FR, pour la Belgique, nous avons fr-BE, et pour les Etats-Unis, nous avons en-US. Nous allons donc créer un dossier nommé fr-FR et un dossier nommé en-US. Nous allons alors déplacer le fichier dvp.html dans le dossier fr-FR, puis en faire une copie que nous collerons dans en-US. Nous l'éditerons alors pour changer "Rechercher" par "Search". Notre gadget pourra alors être en français ou en anglais selon la langue par défaut du système.

Note: Tous les fichiers n'ont pas besoin d'être localisés. Dans notre cas, le fichier JavaScript, les images et même le fichier settings.html n'ont pas besoin d'être changés quelle que soit la langue de l'utilisateur. Ces fichiers peuvent rester en un unique exemplaire à la racine du gadget. Pas besoin de changer les chemins dans le fichier dvp.html, le moteur de gadgets ira les chercher tout seul.

5.2. L'effet dockable

Vous ne l'avez peut-être pas remarqué, mais les gadgets peuvent être placés sur le bureau, simplement par glisser-déposer. A ce moment là, ils peuvent soit rester dans le même état que lorsqu'ils étaient accrochés à la Sidebar, soit changer totalement de forme.

Nous allons donc définir deux états pour notre gadget: l'état docké et l'état non docké. Pour l'état docké, c'est simplement l'état dans lequel nous l'avons développé jusqu'à maintenant. Pour l'état non docké, nous allons agrandir le gadget et lui donner un fond et un contour. Le fonctionnement interne reste totalement identique.

Dans le fichier dvp.js, déclarons deux méthodes que nous accrocherons aux évènements de dock et de undock.

dvp.js
```
System.Gadget.onUndock = WhenUndocked;
System.Gadget.onDock = WhenDocked;
```

Créons alors la fonction pour l'état non docké:

dvp.js
```
function WhenUndocked(){
   System.Gadget.beginTransition();
   with(document.body.style){
        marginTop ="30px";
        marginLeft ="30px";
        width=383;
        height=197;
        backgroundImage="url('../images/fond.png')";
    }

    with(lblRechercher.style){
        fontSize="17px";
    }

    with(txtSearch.style){
        fontSize="17px";
        width = "290px";
    }
    System.Gadget.endTransition(System.Gadget.Transitio
       nType.none,0);
}
```

Voyons ce que signifie le code. Comme vous l'avez deviné, la méthode WhenUndocked() est appelée chaque fois que le gadget est pris de la Sidebar pour être déposé sur le bureau.
Sinon voici la signification des modifications:
document.body.style: pour le gadget en lui-même, je lui définis de nouvelles dimensions (383x197) et je change son fond. Puis je définis une marge à gauche et en haut pour recentrer les éléments (label, Textbox, bouton)
lblRechercher.style: pour le label, j'agrandis sa police
txtSearch.style: pour la Textbox, je redéfinis sa largeur ainsi que la taille du texte. Voici maintenant la méthode lorsque le gadget est replacé sur la Sidebar

dvp.js
```
function WhenDocked(){
        System.Gadget.beginTransition();
        with(document.body.style){
                width=130,
                 height=85,

        backgroundImage="url('../images/bg.gif')";
                marginTop ="3px";
                 marginLeft ="5px";
        }
        with(lblRechercher.style){
                fontSize="13px";
        }

        with(txtSearch.style){
                fontSize="13px";
                width = "120px";
        }

        with (imgSearch.style){
                width = "16px";
        }

         System.Gadget.endTransition(System.Gadget.Trans
itionType.none,0);
}
```

Pas grand chose à expliquer cette fois-ci. Nous redonnons les valeurs d'origine. Voici une image montrant les deux états de notre Gadget.

6. Conclusion

Comme vous avez pu le voir, il n'est pas bien compliqué de développer son propre Gadget et vous n'avez pas d'excuse de ne pas avoir le gadget de vos rêves à partir du moment où vous acceptez de mettre les mains dans le cambouis.

N'oubliez pas, si votre gadget est fonctionnel, n'hésitez pas à le mettre en ligne sur le site MicrosoftGadget, il peut servir à d'autres personnes ;)

Retrouvez l'article de Louis-Guillaume Morand en ligne : Lien48

Vu dans la FAQ

Comment changer l'action du bouton éteindre du menu Démarrer?

Lorsque vous ouvre le menu démarrer vous avez deux boutons: un bouton de mise en veille et un bouton menu qui propose de mettre en veille, éteindre, redémarer, verrouiller, etc.

Peut-être que comme moi, vous désirez ne plus mettre en veille mais avoir le bouton éteindre directement accessible.
Rien de plus simple, allez dans le panneau de configuration > Gestion d'énergie (Power Options) > cliquez sur le premier lien Changer les paramètres du plan d'énergie.
Cliquez alors sur changer les paramètres avancés d'énergie
Dans la nouvelle fenêtre, placez vous sur boutons et diodes

d'énergie(Power buttons & lid). Aller ensuite dans Bouton du menu démarrer et choisissez pour les deux cas: Eteindre, au lieu de mettre en veille. Sauvegarder tout.
Le menu démarrer devrait avoir changé et vous pourrez éteindre plus rapidement votre ordinateur.

Comment créer des raccourcis explorer vers ses emplacements préférés?

Lors que vous vous trouvez dans l'explorateur, vous avez sur un partie gauche un panneau qui contient une arborescence des disques dur, vous permettant d'aller chercher le repertoire où vous désirez vous rendre. Juste au dessus de cette arborescence, se trouve vos favoris explorer. Basé sur le même principe qu'un favori Internet, ces raccourcis vous permettent de vous rendre là où vous le désirez.
Si vous désirez ajouter un raccourci vers un dossier que l'on nommera pour l'exemple "Dossier1", naviguez jusqu'au dossier parent de Dossier1, faites le simplement glisser vers la partie Favoris. Cela aura pour effet de créer un raccourci. Vous pourrez ainsi l'atteindre plus facilement.

Comment paramétrer Windows Defender pour tous les utilisateurs de la machine?

Windows Defender est un logiciel anti-spyware installé de base sur Windows Vista. Celui-ci permet de surveiller les spywares qui tentent de s'installer et d'enlever les éxistants.
Contenant de nombreuses fonctionnalités et paramètres, il vous sera peut-être demandé de le configurer à l'identique pour tous les utilisateurs de l'ordinateur. Pour cela, ouvrez l'éditeur de stratégies (Démarrer > Executer > taper gpedit.msc) et rendez vous à Local Computer Policy \ Computer Configuration \ Administrative Templates \ Windows Components \ Windows Defender

Vous avez alors 6 paramètres que vous pouvez configurer comme bon vous semble:
- Check for New Signatures Before On-Demand Scans
- Turn off Scanning for Spyware and Other Unwanted Software
- Stop Logging Known Good Detections
- Stop Logging Unknown Detection
- Turn off Real-Time Protection Prompts for Unknown Detection
- Download Entire Signature Set

Comment faire pour que Windows Vista ouvre automatiquement les fenêtres qui étaient ouvertes lors de l'extinction de l'ordinateur ?

Cette fonction très pratique faisait par défaut partie des versions de Windows plus anciennes. Il est possible de la réactiver sous Windows Vista en ouvrant un dossier puis Menu Outils > Options des dossiers > Afffichage > Restaurer les fenêtres de dossiers ouvertes lors de la prochaine ouverture de session

Il est également de faire la manipulation directement dans le registre en cliquant sur Démarrer > Exécuter, puis en tapant "regedit" sans les guillemets. Développez l'arborescence HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\Advanced et créez une valeur Dword nommée : PersistBrowsers. Affectez-lui la valeur: 1.

Retrouvez la FAQ Vista en ligne : Lien49

Vu sur les blogs

La RTM de Windows Vista est là

Tout le monde s'y attendait mais c'est maintenant officiel car Jim Allchin vient de poster la nouvelle sur le blog dédié à Windows Vista

La version RTM (Release To Manufacturing) de Windows Vista est là:

http://windowsvistablog.com/blogs/windowsvista/archive/2006/11/08/it-s-time.aspx

Elle devrait donc être normalement bientôt disponible sur MSDN

Bravo à toute l'équipe de développement !

Enjoy !

A+

Retrouvez le bille de Thomas Lebrun en ligne :
http://blog.developpez.com/index.php?blog=9&title=title_25&more=1&c=1&tb=1&pb=1

Utiliser Windows Media Player en VB et VBA

Ce document présente l'utilisation de Windows Media Player en VB et VBA.
La bibliothèque WMP permet de lire des fichiers audio et vidéo, de gérer des PlayList, mais aussi de récupérer des informations sur les lecteurs multimédias de votre poste.

1. Introduction

Tous les exemples présentés dans ce document ont été testés à partir de Windows Media Player 10.0

Les méthodes et propriétés sont identiques pour gérer les fichiers audio ou video.

Les termes employés dans ce document:

Une séquence: est un fichier audio ou une video spécifique
Une PlayList: est un ensemble de séquences.

2. Gérer une séquence

Vous devez préalablement créer une forme dans votre projet et y ajouter un objet Windows Media Player, nommé WindowsMediaPlayer1

2.1. Définir et lire un fichier

Vb
```
'Définit et lit un fichier
WindowsMediaPlayer1.URL = "C:\leFichier.mp3"
```

Cette première procédure charge le fichier et joue automatiquement la séquence.

La propriété autoStart a une valeur True par défaut.
Si vous souhaitez uniquement charger le fichier sans démarrer la lecture, utilisez:

Vb
```
'Désactive la lecture automatique
WindowsMediaPlayer1.settings.autoStart = False

'Définit le fichier
WindowsMediaPlayer1.URL = "C:\leFichier.mp3"
```

Ensuite, pour lire le fichier, utilisez:

Vb
```
WindowsMediaPlayer1.Controls.Play
```

Il est possible de spécifier une position précise dans la séquence pour débuter la lecture.

Vb
```
Private Sub CommandButton1_Click()
    'Chargement fichier & lecture
    WindowsMediaPlayer1.URL = "C:\maMusique.mp3"

    'Positionnement à la 3eme minute
    WindowsMediaPlayer1.Controls.currentPosition = 180
        'secondes
End Sub
```

2.2. Arrêter la lecture

Vb
```
WindowsMediaPlayer1.Controls.Stop
```

2.3. Afficher la durée de la séquence en cours

Vb
```
Private Sub CommandButton1_Click()
    'Lance la lecture du fichier
    WindowsMediaPlayer1.URL = "C:\leFichier.mp3"

    'La récupération d'informations sur une séquence
        peut se faire uniquement
        'lorsque le statut de chargement "Transitioning"
        est atteint
    '9 = statut "Préparation nouvelle séquence"
        (constante wmppsTransitioning)
    While WindowsMediaPlayer1.playState = 9: DoEvents:
        Wend

    'Affiche la durée de la séquence au format hh:mm:ss
    MsgBox
        WindowsMediaPlayer1.currentMedia.durationString
End Sub
```

Remarque:
La propriété Duration renvoie la durée en secondes

Vb
```
'Affiche la durée de la séquence en secondes
MsgBox WindowsMediaPlayer1.currentMedia.duration
```

Pour afficher la position de la lecture en cours au format hh:mm:ss, utilisez:

Vb
```
MsgBox
WindowsMediaPlayer1.Controls.currentPositionString
```

2.4. Effectuer une pause

Vb
```
WindowsMediaPlayer1.Controls.Pause
```

Et pour relancer la séquence:

Vb
```
WindowsMediaPlayer1.Controls.Play
```

2.5. Vérifier si une action peut etre appliquée

La propriété isAvailable permet de contôler si une action peut être effectuée, par exemple vérifier si la séquence peut être mise en Pause.

Vb
```
'vérifie si l'action "pause" peut etre appliquée
(Renvoie Vrai ou Faux)
MsgBox
WindowsMediaPlayer1.Controls.isAvailable("Pause")
```

D'autres actions contrôlables:

- Stop
- Play
- Pause
- FastForward
- FastReverse
- Next
- Previous
- CurrentItem
- CurrentPosition

2.6. Afficher le statut de Windows Media Player

Première solution:

Vb
```
MsgBox WindowsMediaPlayer1.Status
```

Une deuxième solution:

Vb
```
Select Case WindowsMediaPlayer1.playState
    Case 0: MsgBox "Undefined"
    Case 1: MsgBox "Stopped"
    Case 2: MsgBox "Paused"
    Case 3: MsgBox "Playing"
    Case 4: MsgBox "ScanForward" 'Avance rapide
    Case 5: MsgBox "ScanReverse" 'Retour rapide
    Case 6: MsgBox "Buffering"
    Case 7: MsgBox "Waiting"
    Case 8: MsgBox "MediaEnded"
    Case 9: MsgBox "Transitioning" 'Préparation
       nouvelle séquence
    Case 10: MsgBox "Ready"
    Case 11: MsgBox "Reconnecting"
End Select
```

2.7. Désactiver et Réactiver le son

Vb
```
'Désactive le son
WindowsMediaPlayer1.settings.mute = True
```

Vb
```
'Active le son
WindowsMediaPlayer1.settings.mute = False
```

2.8. Modifier le volume sonore

Attribuez une valeur entre 0 et 100 (%) pour modifier la position du curseur de réglage sonore.

Vb
```
'Modifie le volume sonore
WindowsMediaPlayer1.settings.volume = 50
```

2.9. Effectuer une avance rapide

Vb
```
'Effectue une avance rapide
If
WindowsMediaPlayer1.Controls.isAvailable("FastForward")
Then _
   WindowsMediaPlayer1.Controls.fastForward
```

Pour revenir en lecture normale

Vb
```
'pour revenir à la lecture normale
WindowsMediaPlayer1.Controls.Play
```

2.10. Lire la même séquence en boucle

Vb
```
WindowsMediaPlayer1.Controls.Play
WindowsMediaPlayer1.settings.setMode "loop", True
```

Pour vérifier le statut, utilisez:

Vb
```
MsgBox WindowsMediaPlayer1.settings.getMode("loop")
```

2.11. Afficher Windows Media Player en mode plein écran

Windows Media player doit avoir le statut "Lecture en cours" pour utiliser cette option.
Vous pouvez identifier le statut du lecteur en utilisant l'évenement PlayStateChange.

Vb
```
Private Sub WindowsMediaPlayer1_PlayStateChange(ByVal
NewState As Long)
    If NewState = 3 Then _
    WindowsMediaPlayer1.fullScreen = True
End Sub
```

2.12. Afficher des informations sur la séquence en cours

Remarque:

La récupération d'informations sur une séquence peut se faire uniquement lorsque le statut de chargement "Transitioning" est atteint.

Vb
```
MsgBox
WindowsMediaPlayer1.currentMedia.getItemInfo("Name")
'MsgBox
WindowsMediaPlayer1.currentMedia.getItemInfo("author")
'MsgBox
WindowsMediaPlayer1.currentMedia.getItemInfo("Title")
'MsgBox
WindowsMediaPlayer1.currentMedia.getItemInfo("Album")
'MsgBox
WindowsMediaPlayer1.currentMedia.getItemInfo("copyright
")
'MsgBox
WindowsMediaPlayer1.currentMedia.getItemInfo("Artist")
'MsgBox
WindowsMediaPlayer1.currentMedia.getItemInfo("Genre")
'MsgBox
WindowsMediaPlayer1.currentMedia.getItemInfo("Bitrate")
/ 1000 & " kbps"
'MsgBox
WindowsMediaPlayer1.currentMedia.getItemInfo("Abstract"
)
'MsgBox
WindowsMediaPlayer1.currentMedia.getItemInfo("bitRate")
'MsgBox
WindowsMediaPlayer1.currentMedia.getItemInfo("duration"
```

```
)
```

Une autre solution qui liste toutes les informations:

Vb
```
Private Sub CommandButton1_Click()
    Dim Resultat As String
    Dim i As Integer
    Dim Cm As WMPLib.IWMPMedia
    Set Cm = WindowsMediaPlayer1.currentMedia

    'boucle sur les attributs
    For i = 0 To Cm.attributeCount - 1
        If Cm.getItemInfo(Cm.getAttributeName(i)) <> ""
Then _
        Resultat = Resultat & Cm.getAttributeName(i) &
" : " & _
        Cm.getItemInfo(Cm.getAttributeName(i)) & vbLf
    Next

    MsgBox Resultat
End Sub
```

2.13. Modifier les informations d'une séquence

Par exemple pour modifier le titre de la séquence en cours, utilisez:

Vb
```
'Modifie le titre de la séquence en cours
WindowsMediaPlayer1.currentMedia.setItemInfo "title",
"Nouveau_Titre"

'Vérifie le contenu du titre
MsgBox
WindowsMediaPlayer1.currentMedia.getItemInfo("title")
```

3. Gérer une PlayList

Vous devez préalablement créer une forme dans votre projet et y ajouter un objet Windows Media Player nommé WindowsMediaPlayer1

3.1. Créer une PlayList

Cet exemple crée une PlayList composée de 3 séquences:

Vb
```
Dim Xwmp As IWMPMedia

'nettoye la Playlist en cours avant de réalimenter la
liste
WindowsMediaPlayer1.currentPlaylist.Clear

Set Xwmp = WindowsMediaPlayer1.newMedia("C:\essai.mid")
WindowsMediaPlayer1.currentPlaylist.insertItem 0, Xwmp

Set Xwmp =
WindowsMediaPlayer1.newMedia("C:\maMusique.mp3")
WindowsMediaPlayer1.currentPlaylist.insertItem 1, Xwmp

Set Xwmp =
WindowsMediaPlayer1.newMedia("C:\Jumbalaya.mid")
WindowsMediaPlayer1.currentPlaylist.insertItem 2, Xwmp

'Lecture
WindowsMediaPlayer1.Controls.Play
```

3.2. Passer à la séquence suivante ou précédente

Vb
```
'Permet d'accéder à la séquence suivante
WindowsMediaPlayer1.Controls.Next
```

Remarque:
Le premier item de la PlayList sera réactivé si vous êtes arrivé en fin de liste.

Vb
```
'Permet d'accéder à la séquence précédente
WindowsMediaPlayer1.Controls.Previous
```

Remarque:
le dernier item de laPlayList sera activé si vous êtes en début de liste.

3.3. Supprimer une séquence dans la PlayList

Exemple pour supprimer la deuxieme séquence. (L'index du premier élément = 0)

Vb
```
Dim It As Object
Set It = WindowsMediaPlayer1.currentPlaylist.Item(1)
WindowsMediaPlayer1.currentPlaylist.RemoveItem It
```

3.4. Lire une séquence spécifique de la PlayList

Exemple pour lire la 3eme séquence:

Vb
```
Dim It As Object
Set It = WindowsMediaPlayer1.currentPlaylist.Item(2)
WindowsMediaPlayer1.Controls.playItem It
```

3.5. Lister le nom des séquences contenues dans la PlayList

Vb
```
Sub Lister_NomDesSequences_DansLaPlayList()
    Dim Pl As IWMPPlaylist
    Dim j As Integer, i As Integer

    'Définit la playlist active
    Set Pl = WindowsMediaPlayer1.currentPlaylist

    'compte le nombre de séquences dans la playList
    j = Pl.Count
    If Not j > 0 Then MsgBox "il n'y a pas d'éléments
dans la playlist"

    'Boucle sur les séquences
    For i = 0 To j - 1
    MsgBox Pl.Item(i).Name
    '
    'pour afficher la source :
    'MsgBox Pl.Item(i).sourceURL
    '
    Next i
End Sub
```

3.6. Ajouter une séquence dans la PlayList

La séquence est automatiquement placée à la suite des items existants.

Vb
```
Dim Ad As IWMPMedia
Set Ad =
WindowsMediaPlayer1.newMedia("C:\couldntStandTheWeather
.mp3")
```

```
WindowsMediaPlayer1.currentPlaylist.appendItem Ad
```

3.7. Retrouver l'index de la séquence active

Vb

```
Dim Pl As IWMPPlaylist
Dim j As Integer, i As Integer
Dim Cible As String

Cible = WindowsMediaPlayer1.Controls.currentItem.Name

Set Pl = WindowsMediaPlayer1.currentPlaylist
j = Pl.Count
If Not j > 0 Then MsgBox "il n'y a pas d'éléments dans
la playlist"

For i = 0 To j - 1
    If Cible = Pl.Item(i).Name Then
        MsgBox "L'index de la séquence " & Cible & "
est : " & i
        Exit For
    End If
Next i
```

3.8. Compter le nombre de séquences dans la PlayList

Vb

```
WindowsMediaPlayer1.currentPlaylist.Count
```

4. Gérer les CD Audio

Vous devez préalablement créer une forme dans votre projet et y ajouter un objet Windows Media Player nommé WindowsMediaPlayer1

L'exemple suivant montre comment définir le contenu d'un CD audio (Tracks) en tant que PlayList.

Vb

```
Private Sub CommandButton1_Click()
     Dim Lecteur As wmplib.IWMPCdrom
     Dim It As wmplib.IWMPMedia

    'Définit le lecteur contenant le CD
    'Item(0) Correspond au 1er lecteur du PC
    Set Lecteur =
WindowsMediaPlayer1.cdromCollection.Item(0)

    'Définit le contenu du lecteur en tant que PlayList
    WindowsMediaPlayer1.currentPlaylist =
Lecteur.Playlist

    '----
    'à partir de ce point, la lecture est lancée
automatiquement.
    '----
    'Les lignes suivantes montrent simplement comment
atteindre
    'une séquence spécifique.
    '----

    'Permet de vérifier si WMP est prêt
    If WindowsMediaPlayer1.playState = 0 Then
        MsgBox "Opération annulée."
        Exit Sub
    End If

    'Active la 3eme séquence du CD pour la lecture
    Set It =
WindowsMediaPlayer1.currentPlaylist.Item(2)
    WindowsMediaPlayer1.Controls.playItem It
End Sub
```

Les méthodes et propriétés sont identiques aux exemples présentés dans les chapitres précédents.

5. Utiliser la librairie sans objet support

Pour les fichiers musicaux, il est possible de lancer une séquence ou de gérer une playlist sans ajouter d'objet dans votre projet, en utilisant directement la bibliothèque WindowsMediaPlayer (wmp.dll).

Pour activer la référence en VBA:
Dans l'éditeur de macros,
Menu Outils
References
Cochez la ligne "Windows Media Player"
Cliquez sur OK pour valider

Pour activer la référence en VB:
Menu Projet
Références
Cochez la ligne "Windows Media Player" (wmp.dll)
Cliquez sur OK pour valider

Vb

```
Option Explicit
Dim Wmp As WindowsMediaPlayer

Sub jouerWindowsMediaPlayer()
        Set Wmp = CreateObject("WMPlayer.OCX.7")
        Wmp.URL = "C:\leFichier.mp3"
        Wmp.Controls.Play
End Sub
```

et pour arrêter la séquence

Vb

```
Sub arreterWondowsMediaPlayer()
    If Wmp Is Nothing Then Exit Sub
    Wmp.Controls.stop
End Sub
```

Les méthodes et propriétés sont identiques aux exemples présentés dans les chapitres précédents.

6. Divers

6.1. Lister les lecteurs de CD et de DVD installés

La référence WindowsMediaPlayer (wmp.dll) doit être préalablement activée.

Vb

```
Sub listeLecteurs_CD_DVD()
    Dim Wmp As WindowsMediaPlayer
    Dim nombreLecteurs As Integer, i As Integer
    Dim PCd As IWMPCdrom

    Set Wmp = CreateObject("WMPlayer.OCX.7")

    nombreLecteurs = Wmp.cdromCollection.Count
    If Not nombreLecteurs > 0 Then _
    MsgBox "il n'y a pas de lecteur de CD ou DVD
installés ."

    For i = 0 To nombreLecteurs - 1
        Set PCd = Wmp.cdromCollection.Item(i)
```

```
        MsgBox PCd.driveSpecifier
    Next i
End Sub
```

5.2. Ouvrir le lecteur de CD / DVD

Vb

```
Sub ouvrirLecteur()
    Dim Wmp As Object, Lecteur As Object

    Set Wmp = CreateObject("WMPlayer.OCX.7")

    'L'index 0 est correspond au 1er lecteur du poste
    Set Lecteur = Wmp.cdromCollection.Item(0)
    Lecteur.eject
End Sub
```

5.3. Lancer le lecteur Windows Media Player pour lire un fichier

Vb

```
Sub lancerUneVideo()
        'Necessite d'activer la référence Windows Media
Player
        Dim Wmp As WindowsMediaPlayer

        Set Wmp = CreateObject("WMPlayer.OCX.7")
        Wmp.openPlayer "C:\monFilm.mpg"
End Sub
```

Retrouvez l'article de SilkyRoad en ligne : Lien50

Vu dans la FAQ VBA

Comment programme-t-on la correction d'orthographe ?

Cela dépend de ce que l'on veut obtenir. Pour démarrer la correction sur une feuille par exemple il suffit de faire :

```
Application.CheckSpelling
CustomDictionary:="PERSO.DIC", IgnoreUppercase:=False,
AlwaysSuggest:=True
```

Ceci a pour effet d'ouvrir la fenêtre du correcteur. Il est possible de contrôler un seul mot, avec une réponse booléenne.

```
Private Function MotExiste(ByVal strMot As String) As
Boolean
    MotExiste = Application.CheckSpelling(strMot,
"PERSO.DIC", False)
End Function
```

Comment écrit-on une fonction de feuille de calcul ?

Il s'agit d'une fonction normale. Par exemple

```
Function InvChaine(Cellule As Range) As Variant

Dim MaChaine As String, cmpt As Long

If Len(Cellule.Value) = 0 Then
    InvChaine = CVErr(xlErrValue)
Else
    MaChaine = CStr(Cellule.Value)
    For cmpt = Len(MaChaine) To 1 Step -1
        InvChaine = InvChaine & Mid(MaChaine, cmpt, 1)
    Next cmpt
End If

End Function
```

Si dans votre feuille vous écrivez la formule =invchaine(E2) La cellule contiendra la chaine retournée ou #VALEUR

Comment savoir si une plage fait référence à une cellule ou a plusieurs, voire à plusieurs plages ?

Il faut utiliser la propriétés Count des collections visées. C'est la collection Cells pour le nombres de cellules, Areas pour le nombre de plages. Un code Exemple pourrait être.
Dim Msg As String, objRange As Range

```
With
ThisWorkbook.Worksheets(1).Range("A3:A11,D7:E14,G1:G4,I
15:J23,C22:F22")
    Msg = "La plage contient " & .Cells.Count & "
cellules dans " & .Areas.Count & " plages continues" &
vbCrLf
    For Each objRange In .Areas
        Msg = Msg & "La plage " &
objRange.AddressLocal(False, False, xlA1) & " contient
" & objRange.Cells.Count & " cellules" & vbCrLf
    Next
End With
MsgBox Msg
```

Retrouvez la FAQ VBA en ligne : Lien51

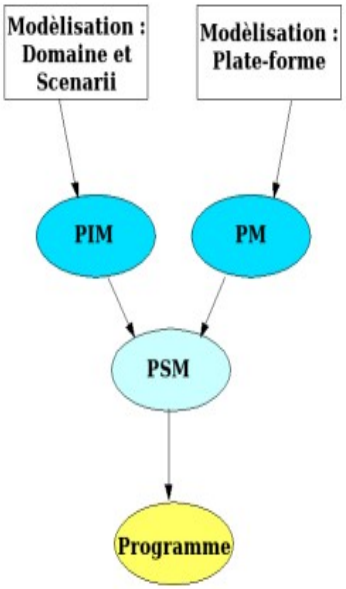
Les derniers tutoriels et articles

Introdution au MDA

Introduction générale au développement orienté modèle

1. Introduction

l s'agit de modéliser l'application que l'on veut créer de manière indépendante de l'implémentation cible (niveau matériel ou logiciel). Ceci permet une grande réutilisation des modèles.



Les modèles ainsi créés (PIM - Platform Independant Model) sont associés à des modèles de plate-forme (PM - Platform Model), et transformés, pour obtenir un modèle d'application spécifique à la pate-forme (PSM - Platform Specific Model).

Des outils de génération automatique de code permettent ensuite de créer le programme directement à partir des modèles.

Cette approche permet de plus de faire évoluer facilement, à partir des modèles, les applications : le développement d'un nouveau module, quand tous les modèles nécessaires sont disponibles, peut ne pas prendre plus de quelques minutes.

2. Les outils de MDA

Pour obtenir une telle efficacité, plusieurs outils conceptuels sont mis à disposition. La technologie MDA (Model Driven Architecture) est supportée par l'OMG (Object Management Group), qui propose également UML (Unified Modeling Language) et Corba (Object Request Broker).

Ces outils sont :

- **UML**, largement utilisé par ailleurs, qui permet une mise en oeuvre aisée de MDA en offrant un support connu,
- **XMI**, XML Metadata Interchange, qui propose un formalisme de structuration des documents XML de telle sorte qu'ils permettent de représenter des méta-données d'application de manière compatible,
- **MOF**, Meta Object Facility, spécification qui permet le stockage, l'accès, la manipulation, la modification, de méta-données,
- **CWM**, base de données pour méta-données.

L'OMG n'a pas jugé utile de standardiser un processus associé à ces outils. Leur rôle est de répondre aux besoins des utilisateurs de manière générique, et non de proposer de solutions définitives pour certains types d'applications précises.

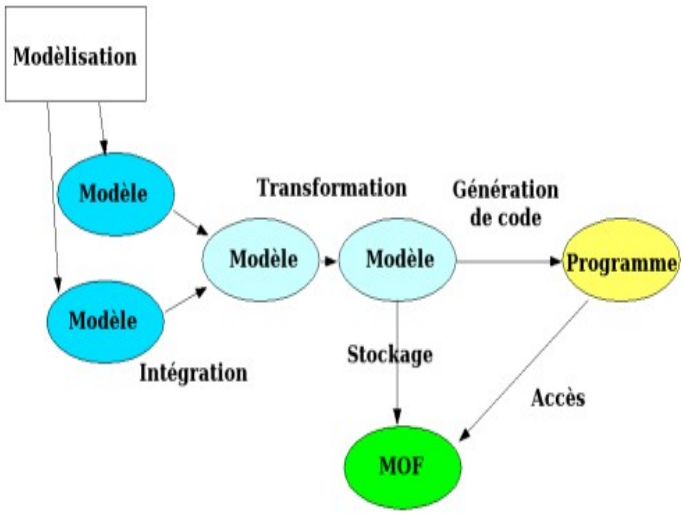
Un processus de génie logiciel exploitant les possibilités de MDA a cependant été proposé : le 'Model-Driven Software Development' (Lien52).

3. MDA dans la pratique

Un certain nombre d'étapes et de transformation sont identifiables dans un processus MDA :

- modélisation UML,
- transformation intermédiaires,
- génération de code.

Cette suite de transformations est illustrée par l'image suivante :



3.1. La modélisation

C'est une étape préalable. Elle se fait en utilisant les outils

conceptuels d'UML (Unified Modeling Language). Elle aboutit à un ensemble de modèles de la future application, représentés par des diagrammes de classes. Chaque domaine fonctionnel de l'application est représenté indépendamment des autres.

A partir de ces diagrammes de classes UML, des modèles manipulables sont générés (au format XMI).

3.2. L'intégration

Plusieurs modèles sont utilisés pour la création d'une application. Typiquement, il s'agit de modèles représentant des fonctionnalités différentes. Ils sont assemblés en un seul modèle, qui représente l'application, et sont à partir de là manipulables comme un unique modèle.

3.3. La transformation

A partir de modèles génériques, il s'agit de préciser ce que sera l'application : format de données, réalisation des fonctionnalités. Le(s) modèle(s) de l'application sont donc complétés, affinés.

Typiquement, il s'agit à ce niveau là de transformer un méta-modèle (modèle de modèle, qui indique les contraites que doit respecter l'application), en modèle fonctionnel, dotés de services particuliers.

Il peut également s'agir de transformer un modèle fonctionnel indépendant de l'application (PIM - Platform Independant Model) en modèle prennant en compte les contraintes de déploiement (PSM - Platform Specific Model).

3.4. La génération de code

Lorsque le modèle est complet, le code est généré. L'application peut alors être déployée.

Dans la pratique, le code généré doit être complété : l'implémentation des différentes méthodes, par exemple, n'est pas explicitée dans le modèle.

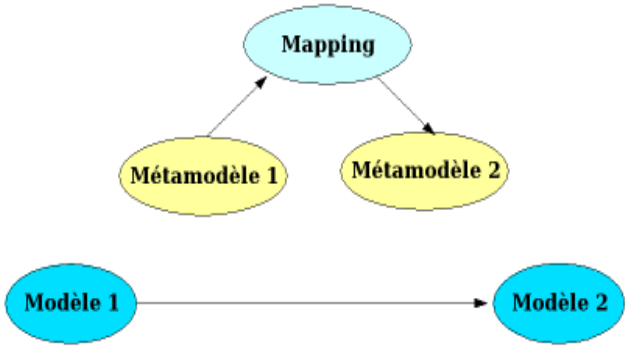
En cas d'extension ultérieure du modèle, il est indispensable de disposer d'un environnement de développement qui conserve le code ajouté. Si ce n'est pas le cas, le développement incrémental est rendu beaucoup plus laborieux, et la création de grandes applications est compromise.

3.5. Le stockage et l'accès

Les modèles peuvent être réutilisés pour le développement d'autres applications, mais ils sont également accessibles par les applications elles-même. La spécification MOF (Metadata Object Facility) permet de définir des bases de données de modèles accessibles de manière transparente par les applications.

4. Les Transformations

La transformation de modèle MDA se fait par mapping entre un modèle initial et un modèle cible. Chaque modèle doit être décrit par un méta-modèle, qui recense les caractéristiques de ce modèle. Le mapping est alors définit comme une traduction entre le méta-modèle initial et le méta-modèle cible. La figure suivante présente le principe de la transformation.



Pour effectuer la transformation, un moteur de transformation est indispensable.

4.1. Les Méta-modèles

Les méta-modèles sont exprimés couramment sous format XMI. Ils peuvent représenter un langage connu, par exemple UML ou Java. Ils peuvent également être conçus spécifiquement pour un domaine d'application donné.

Deux approches existent pour créer un langage de méta-modèle :

- **MOF** : On crée à partir de rien un langage nouveau, en respectant les spécifications du MOF (Meta Object Facility),
- **UML** : On crée une extension d'UML, à l'aide de stéréotypes, de contraintes (langage OCL).

4.1.1. Les niveaux de méta-modèles

On définit quatre niveaux de méta-modèles :

- le niveau zéro est l'implémentation du programme,
- le niveau 1 est le modèle du programme (classes)
- le niveau 2 est le méta-modèle du programme, c'est à dire le langage
- le niveau 3 est le méta-modèle des méta-modèles, ou méta-méta-modèle, c'est à dire la définition d'un méta-modèle. Il s'agit donc de la spécification MOF.

Il faut noter que les niveaux ne sont pas absolu : en cas de nécessité, on peut avoir plusieurs modèles intermédiaires.

M3	MOF – Meta modèle de définition de langage
M2	Modèle de de langage
M1	Classes d'un programme
M0	Instanciation

4.2. Les Mappings

On distingue deux types de mappings : les mapping verticaux, qui changent le niveau d'abstraction, et les mapping horizontaux, qui le conservent.

IV-B-1. Les mappings verticaux

On en distingue deux sortes :

- Mappings de modèle d'analyse vers le modèle d'implémentation (ou mappings de rafinnement). Ils permettent de préciser le modèle de l'application, en fonction de l'implémentation souhaitée. Souvent, un seul modèle initial suffit. Parfois, des contraintes sont nécessaires (sinon le modèle cible est incomplet), ou bien plusieurs modèles sont utilisés comme source.
- Mappings d'abstraction. C'est la transformation inverse. Elle permet une meilleure compréhension du code (reverse engineering), ou la migration d'une plate-forme vers une autre.

IV-B-2. Les mappings horizontaux

- Mappings de représentation. On l'utilise pour passer d'un format à un autre, le second disposant de plus d'outils de manipulation et de représentation. Peu utile s'ils ne sont pas reversibles.
- Mappings d'optimisation, pour améliorer la performance.
- Mappings de reconstruction, pour améliorer la maintenabilité et la lisibilité du code.

Les mappings sont réalisés soit par des règles générales, soit par corrélation, c'est à dire par appariement de propriétés du modèle source avec des propriétés du modèles cible.

5. Une Méthode

Maintenant que les outils et les principes de MDA ont été présentés, voici une méthode de développement d'application correspondante. Elle a pour vocation d'améliorer la productivité, c'est à dire le temps de création de la première version de l'application, de même que l'évolutivité, c'est à dire la possibilité d'améliorer et d'étendre cette application.

On distinguera une approche linéaire, exploitable pour des petites applications ou des sous-systèmes, et une approche incrémentale, qui doit permettre de mettre en oeuvre des applications de plus grandes dimensions.

5.1. Approche linéaire

Phase de spécifications
- choix des principes de l'application (et donc des modèles)
- adaptation de ces modèles
- validation de l'architecture au niveau fonctionnel

Phase de design
- raffinement automatique (intégration de modèles existants)
- raffinement manuel

Phase d'implémentation
- génération de code pour la plate-forme-cible
- complétion du code généré
- intégration d'outils pré-existants

Phase de validation
- bon fonctionnement de l'application
- validation des spécifications

5.2. Approche incrémentale

Cette approche met en évidence la puissance de MDA. L'évolution permanente du logiciel est intégrée dans la méthodologie de développement, ce qui a deux avantages principaux :

- lors de la première réalisation d'une application, des versions fonctionnelles intermédiaires sont disponibles, ce qui permet d'accélérer la mise en production - éventuellement d'une version incomplète, préférable à un retard pur et simple,
- lors de l'évolution d'une application, la méthodologie est conservée, et la validation est donc facilitée.

Prototype

Réalisation d'une architecture comportant les fonctionnalités minimales (IHM et fonctionnalités clés) selon la méthode linéaire.

Architecture complète

Intégration des différents principes de fonctionnement : type de client (lourd/léger), support de mobilité, etc., selon la méthode linéaire.

On obtient un squelette d'application, comportant tous ses éléments, mais sans implémentation.

Mise en place des fonctionnalités

Compléter chaque fonctionnalité l'une après l'autre, selon la méthode linéaire.

Récursivité

Les étapes 2, 3 peuvent être réalisés par suite d'affinements successifs. Chaque affinement doit permettre de nouveaux types d'usage (extension des fonctionnalités), les version intermédiaires doivent être exploitables.

Validation de la version finale

Evolutions

Elles peuvent se faire selon le même principe que les affinements succesifs ayant conduit à la version complète du produit.

Retrouvez l'article de Pierre Parrend en ligne :
Lien53

Liens

Lien1 : http://troumad.developpez.com/linux/serveur/xorg
Lien2 : http://kerneltrap.org/node/7440
Lien3 : http://kernelnewbies.org/Linux_2_6_19
Lien4 : http://blog.developpez.com/index.php?blog=79&title=nouvelle_version_du_noyau_linux_2_6_19
Lien5 : https://jdk6.dev.java.net/
Lien6 : https://scripting.dev.java.net/
Lien7 : http://jcp.org/en/jsr/detail?id=223
Lien8 : http://jcp.org/en/jsr/detail?id=199
Lien9 : http://jcp.org/en/jsr/detail?id=269
Lien10 : http://jcp.org/en/jsr/detail?id=250
Lien11 : http://jcp.org/en/jsr/detail?id=202
Lien12 : http://jcp.org/en/jsr/detail?id=221
Lien13 : http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/
Lien14 : http://jcp.org/en/jsr/detail?id=105
Lien15 : http://jcp.org/en/jsr/detail?id=173
Lien16 : http://jcp.org/en/jsr/detail?id=181
Lien17 : http://jcp.org/en/jsr/detail?id=221
Lien18 : http://jcp.org/en/jsr/detail?id=224
Lien19 : http://java.sun.com/javase/6/webnotes/features.html
Lien20 : http://java.sun.com/javase/6/docs/
Lien21 : http://java.sun.com/javase/6/docs/api/
Lien22 : http://www.developpez.net/forums/showthread.php?t=250242
Lien23 : http://adiguba.developpez.com/tutoriels/java/6
Lien24 : http://blog.developpez.com/index.php?blog=40&cat=657
Lien25 : http://php.developpez.com/faq/?page=fichiers_chmod
Lien26 : http://antoine-herault.developpez.com/tutoriels/php/gestionnaire/
Lien27 : http://g-rossolini.developpez.com/tutoriels/php/les-formulaires-et-php5/
Lien28 : http://antoine-herault.developpez.com/tutoriels/php/upload/
Lien29 : http://julien-pauli.developpez.com/afup/forum2006
Lien30 : http://php.developpez.tv/forumafup2006
Lien31 : http://flash.developpez.com/livres
Lien32 : http://www.gotdotnet.com/team/FxCop/
Lien33 : http://webman.developpez.com/articles/dotnet/fxcop
Lien34 : http://odelmotte.developpez.com/tutoriels/dotnet/reporting
Lien35 : http://hulk.developpez.com/tutoriel/crystalreport
Lien36 : http://blog.developpez.com/index.php?blog=36&title=developer_aamp_it_pro_days_2007_gand_bel
Lien37 : http://mdeverdelhan.developpez.com/tutoriel/lua/tutoriel2
Lien38 : http://c.developpez.com/livres
Lien39 : http://xml.developpez.com/cours/
Lien40 : http://khany.developpez.com/tutoriel/xml
Lien41 : http://xml.developpez.com
Lien42 : https://metalink.oracle.com/metalink/plsql/f?p=130:10:1006804008971516608::::alltext,knowledge,numHits:168697.1,TRUE,100
Lien43 : https://metalink.oracle.com/metalink/plsql/f?p=130:14:1006804008971516608::::p14_database_id,p14_docid,p14_show_header,p14_show_help,p14_black_frame,p14_font:NOT,259387.1,1,1,1,helvetica
Lien44 : https://metalink.oracle.com/metalink/plsql/f?p=130:10:1006804008971516608::::alltext,knowledge,numHits:259379.1,TRUE,100
Lien45 : http://fadace.developpez.com/oracle/pwd
Lien46 : http://baptiste-wicht.developpez.com/tutoriel/ms-sql/securiser
Lien47 : http://sgbd.developpez.com/livres/
Lien48 : http://lgmorand.developpez.com/articles/sidebar-gadget
Lien49 : http://windows.developpez.com/faq/vista
Lien50 : http://silkyroad.developpez.com/VBA/WindowsMediaPlayer
Lien51 : http://vb.developpez.com/faqvba
Lien52 : http://www.mdsd.info/
Lien53 : http://pparrend.developpez.com/tutoriel/mda-intro