Developpez Magazine

Edition de Février-Mars 2007.
Numéro 7.
Magazine en ligne gratuit.
Diffusion de copies conformes à l'original autorisée.
Réalisation : Baptiste Wicht
Rédaction : la rédaction de Developpez
Contact : magazine@redaction-developpez.com

Index

Editorial

Developpez.com Magazine revient pour son 8ème numéro avec toujours autant (sinon plus) d'articles, de tutoriels, de FAQ, ...

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

La rédaction

Tutoriel C

Ecriture de Driver en C

Dans cet article, vous verrez les bases pour créer vos propres drivers sous linux grâce au langage C.

par **Benjamin Roux**

Page 22

Article XML

La structure des fichiers OpenXML

Faites connaissance avec l'organisation interne des fichiers OpenXML, le nouveau format de document Office de Microsoft.

par **Eric Grimois**

Page 29

Exécuter une application externe en Java

1. Introduction

Il arrive fréquemment que l'on doive lancer une application externe depuis un programme Java. Java nous le permet, cependant beaucoup de personnes rencontrent des difficultés souvent dues à une méconnaissance de certains principes pourtant fondamentaux.

2. Lancer une application externe

2.1. La classe Runtime

L'exécution d'une application externe se fait grâce aux méthodes exec() de la classe Runtime. Chaque application Java possède une instance unique de la classe Runtime qui lui permet de s'interfacer avec son environnement.

L'instance se récupère avec la méthode statique getRuntime().

```
Runtime runtime = Runtime.getRuntime();
```

Pour lancer votre application externe il vous suffit maintenant d'appeler l'une des six méthodes exec() de la classe Runtime et dont voici les déclarations :

```
public Process exec(String command);
```

Permet d'exécuter une ligne de commande dans un processus séparé.

```
public Process exec(String[] cmdarray);
```

Permet d'exécuter une commande avec ses arguments dans un processus séparé.

```
public Process exec(String[] cmdarray, String[] envp);
```

Permet d'exécuter une commande avec ses arguments dans un processus séparé en spécifiant des variables d'environnement.

```
public Process exec(String[] cmdarray, String[] envp,
File dir);
```

Permet d'exécuter une commande avec ses arguments dans un processus séparé en spécifiant des variables d'environnement et le répertoire de travail.

```
public Process exec(String command, String[] envp);
```

Permet d'exécuter une ligne de commande dans un processus séparé en spécifiant des variables d'environnement.

```
public Process exec(String command, String[] envp, File
dir);
```

Permet d'exécuter une ligne de commande dans un processus séparé en spécifiant des variables d'environnement et le répertoire de travail.

Remarque : Les variables d'environnement spécifiées doivent l'être selon le format nom=valeur.

Un point important est que si vous voulez lancer une application externe en lui passant des paramètres, il faut toujours passer par une des méthodes exec() attendant un tableau de String.

Même s'il est possible d'utiliser une des méthodes exec() attendant un simple String pour lancer une application avec des paramètres :

```
Runtime runtime = Runtime.getRuntime();
runtime.exec("monappli param1 param2");
```

cela risque de poser des problèmes si l'un de vos paramètres contient un espace.

En effet, java utilise le caractère espace pour extraire les différents paramètres de la ligne de commande. Donc si vous avez un paramètre du genre "un paramètre avec des espaces", Java le comprendra comme cinq paramètres différents ("un", "paramètre", "avec", "des", "espaces").

C'est pourquoi il faut toujours utiliser un tableau de String pour passer des paramètres à une application externe.

```
Runtime runtime = Runtime.getRuntime();
runtime.exec(new String[] { "monappli", "un paramètre
avec des espaces", "param2" } );
```

Cette remarque est aussi valable si la commande elle-même contient des espaces.

```
Runtime runtime = Runtime.getRuntime();
runtime.exec(new String[] { "C:\\Program
Files\\MonAppli\monappli.exe" } );
```

2.2. La classe Process

Comme vous l'avez sans doute remarqué, les différentes méthodes exec() renvoie un objet de type Process. Cette classe représente le processus de l'application externe et va nous permettre d'interagir avec lui. La classe Process, qui est abstraite, définit les 6 méthodes suivantes :

- la méthode destroy() qui permet de tuer le processus de l'application externe,
- la méthode exitValue() qui permet de récupérer la valeur de retour du processus de l'application externe,
- la méthode getErrorStream() qui permet de récupérer le flux d'erreur du processus de l'application externe,
- la méthode getInputStream() qui permet de récupérer le flux de sortie du processus de l'application externe,
- la méthode getOutputStream() qui permet de récupérer le flux d'entrée du processus de l'application externe,
- la méthode waitFor() qui met le thread courant en attente que le processus de l'application externe se termine.

3. Communiquer avec l'application

Remarque : pour cette partie il est nécessaire de connaître le fonctionnement des flux d'entrée/sortie en Java (cf ce tutoriel sur le package java.io).

3.1. Récupération des flux

Si besoin est, nous avons la possibilité de communiquer avec notre application externe au travers des trois flux récupérables par les méthodes getErrorStream(), getInputStream() et getOutputStream() de la classe Process :

- la méthode getErrorStream() permet de récupérer un InputStream représentant le flux d'erreur de l'application externe.
- la méthode getInputStream() permet de récupérer un InputStream représentant le flux de sortie de l'application externe.
- la méthode getOutputStream() permet de récupérer un OutputStream représentant le flux d'entrée de l'application externe.

Remarque : au premier abord il peut paraître étrange de récupérer un InputStream pour le flux de sortie standard. Cependant il faut bien se placer au niveau de l'application Java.
En effet, il s'agit de la sortie standard de l'application externe, l'application Java va lire ce flux qui est donc de son point de vue (en fait le notre) un flux d'entrée (idem pour le flux d'erreur). De même pour l'entrée standard de l'application externe, du point de vue de l'application Java il s'agit d'un flux de sortie puisqu'elle y écrit (d'où le OutputStream).

3.2. Consommation des flux

L'un des problèmes majoritairement rencontré est le fait que l'application externe semble se bloquer. Cela est souvent dû à une mauvaise gestion des flux.
En effet, il arrive que l'application externe envoie des données sur les flux de sortie sans que cela soit "visible" (notamment sur le flux d'erreur ou lorsque l'application externe est une interface graphique). Pour palier à ce problème, il est nécessaire de consommer tous les flux de sortie. De plus, chaque flux doit être géré dans un thread différent.

```
Runtime runtime = Runtime.getRuntime();
Process process = runtime.exec("monappli");

// Consommation de la sortie standard de l'application
externe dans un Thread separe
new Thread() {
	public void run() {
		try {
			BufferedReader reader = new
	BufferedReader(new
	InputStreamReader(process.getInputStream()));
			String line = "";
			try {
				while((line =
	reader.readLine()) != null) {
					//Traitement
				}
			} finally {
				reader.close();
			}
		} catch(IOException ioe) {
			ioe.printStackTrace();
		}
	}
}.start();

// Consommation de la sortie d'erreur de l'application
externe dans un Thread separe
new Thread() {
	public void run() {
		try {
			BufferedReader reader = new
	BufferedReader(new
	InputStreamReader(process.getErrorStream()));
			String line = "";
			try {
				while((line =
	reader.readLine()) != null) {
					// Traitement
				}
			} finally {
				reader.close();
			}
		} catch(IOException ioe) {
			ioe.printStackTrace();
		}
	}
}.start();
```

4. Mise en pratique

La classe ProcessLauncher (Lien1) utilise les principes vus précédemment afin de lancer une application externe dans de bonnes conditions.

5. Remarques

5.1. La classe ProcessBuilder

Depuis Java 5, nous avons à notre disposition la classe ProcessBuilder qui permet entre autres de fusionner les flux de sortie et d'erreur du Process.

5.2. JDIC et la classe Desktop

L'API JDIC possède une classe très intéressante pour notre sujet. Il s'agit de la classe Desktop qui permet notamment d'ouvrir un fichier avec l'application qui lui est associée par le système. A noter cependant que vous n'aurez aucun contrôle sur le processus lancé et que vous ne pourrez pas communiquer avec lui.

Pour plus d'information sur l'API JDIC, reportez vous au tutoriel JDesktop Integrated Components.

5.3. Java 6 et la classe Desktop

Depuis Java 6, la classe Desktop a été intégrée dans l'API standard et est quasiment identique à sa soeur de l'API JDIC.

5.4. Les commandes Windows

Une autre remarque concerne les commandes Windows. En effet certaines d'entre elles ne peuvent être lancées comme une application externe (la commande "dir" par exemple). Cela est du au fait qu'elles nécessitent le programme "cmd.exe" pour pouvoir être exécutées.

```
Runtime runtime = Runtime.getRuntime();
runtime.exec(new String[] { "cmd.exe", "/C", "dir" });
```

Retrouvez l'article en ligne de Yann D'Isanto : Lien2

1. Introduction

Java SE 6 offre une version totalement refaite de la classe **SwingWorker**. Cette classe permet de faciliter les interactions entre un thread utilisateur et **l'EventDispatchThread**. Pour utiliser la classe **SwingWorker**, il faut tout d'abord comprendre les problèmes et les motivations d'une telle abstraction. Ce tutoriel va tenter d'expliquer le but de cette classe, et de présenter un exemple d'utilisation.

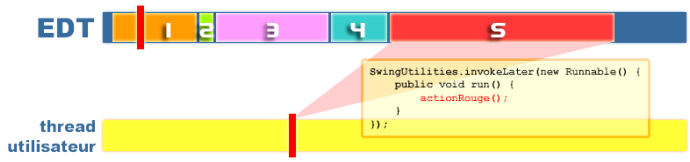
2. L'EventDispatchThread

Lorsque vous démarrez un programme java, 3 threads sont démarrés :

- le thread principal qui exécute la méthode main ;
- le thread de gestion mémoire (garbage collector) ;
- l'EventDispatchThread.

L'EventDispatchThread est le thread qui s'occupe de l'interface graphique, c'est-à-dire l'affichage des fenêtres, les actions à exécuter lors des clics sur les boutons, etc... Le principe est simple : tout ce qui concerne l'interface graphique doit être exécuté dans ce thread, de manière séquentielle. Ceci a un gros avantage : il n'y a pas besoin de synchronisation, puisque les actions de l'interface graphique ne sont pas exécutées en parallèle.

Mais bien sûr, parfois, à partir d'un thread utilisateur, nous avons besoin de modifier des éléments graphiques (mise à jour de textfields, désactivation d'un bouton...). Dans ce cas, il faut utiliser la méthode static **SwingUtilities#invokeLater(Runnable)**, où **Runnable** contient les actions à effectuer dans **l'EDT**.



Comme vous pouvez le voir sur le schéma, le Runnable s'exécutera de manière asynchrone par rapport au thread utilisateur (d'où le nom de la méthode invokeLater). Les curseurs rouges représentent l'exécution courante dans chaque thread. Aussitôt après l'appel à invokeLater dans le thread utilisateur, le code continue de s'exécuter, alors que le Runnable n'est peut-être même pas commencé. Pour attendre la terminaison de l'exécution du Runnable avant de continuer, il faut utiliser SwingUtilities#invokeAndWait(Runnable) (mais dans la majorité des cas, on peut s'en passer).

Certaines méthodes de l'API Swing sont thread-safe, c'est-à-dire que la contrainte d'exécuter cette méthode dans l'EDT n'est pas violée si vous l'appelez depuis un thread utilisateur : c'est la méthode qui se charge de l'exécution dans l'EDT. Cette particularité est précisée dans la javadoc de ces méthodes, mais de toute façon, vous ne perdrez rien à les exécuter dans l'EDT.

Les actions effectuées par l'EDT ne sont pas forcément fournies par l'utilisateur par le biais d'un Runnable. Les méthodes des listeners de Swing sont également appelées dans l'EDT, tout comme tout ce qui concerne l'affichage des fenêtres et son rafraîchissement. Or, si sur le schéma ci-dessus l'action 4 correspond à la mise à jour de l'affichage de la fenêtre, et que l'action 3 représente un calcul très coûteux en temps, l'interface va se figer (en savoir plus sur ce comportement) et vous aurez l'impression que votre ordinateur rame (alors que c'est simplement dû à une mauvaise programmation). C'est pourquoi il est nécessaire d'effectuer des calculs dans des threads utilisateurs, mais tout en respectant la contrainte que l'affichage doit être exécutée dans l'EDT.

Bien sûr, nous pourrions créer un Thread qui effectue les calculs, et qui périodiquement, au fur et à mesure du calcul, met à jour l'interface graphique en appelant invokeLater. C'est faisable, mais c'est justement ce que la classe SwingWorker permet de simplifier.

3. Présentation de SwingWorker

SwingWorker est une classe abstraite, possédant 2 types paramétrés, qui seront décrits lors de la présentation des méthodes les utilisant.

Voici une présentation des méthodes principales à utiliser. (Vous pouvez également consulter la javadoc (Lien3) pour plus d'informations.)

- **protected abstract T doInBackground()** : **à redéfinir**. La seule méthode abstraite. C'est dans cette méthode qu'il faut définir le code à exécuter dans un thread séparé (un calcul long par exemple). Cette méthode retourne un résultat, du type T (type passé en paramètre de la classe), que l'on peut récupérer grâce à la méthode get() une fois le traitement terminé.
- **protected void done()** : **à redéfinir (éventuellement)**. Permet d'effectuer des actions dans l'EDT une fois que le traitement (effectué par doInBackground) est terminé.
- **protected final void publish(V... chunks)** : Permet de transmettre des résultats partiels du traitement. Le paramètre utilise l'ellipse, permettant d'avoir un nombre quelconque d'arguments (en savoir plus sur l'ellipse), ici de type V. V est le second type passé en paramètre de la classe, et définit le type des résultats partiels à transmettre à la méthode process.
- **protected void process(List<V> chunks)** : **à redéfinir (éventuellement)**. Permet de récupérer les résultats partiels du traitement dans l'EDT publiés par la méthode publish. Cette méthode prend en paramètre une liste de V, et non uniquement un V, car pour des raisons d'efficacité, plusieurs appels à publish peuvent résulter en 1 seul appel à process, en transmettant donc plusieurs résultats à la fois.
- **public final T get() :** Permet de récupérer le résultat renvoyé par doInBackground, en attendant éventuellement que le traitement se termine s'il n'est pas terminé. Cette méthode étant bloquante, il faut donc éviter de l'appeler à partir de l'EDT (sauf si vous savez ce que vous faites).
- **protected void setProgress(int progress)** : Indique le nouvel état d'avancement du traitement (à appeler donc dans la méthode doInBackground). Comme nous le verrons plus tard, ceci est bien pratique pour mettre à jour une barre de progression.
- **public final void addPropertyChangeListener (PropertyChangeListener listener)** : Ajoute un écouteur de propriétés, permettant notamment d'écouter l'avancement du traitement, provoqué par setProgress.
- **public final void execute()** : Démarre l'exécution de doInBackground dans un thread séparé.

4. Exemple d'utilisation

Pour mieux comprendre comment fonctionne SwingWorker, étudions un exemple. Nous voulons compter combien de fichiers se trouvent dans le répertoire utilisateur, en comptant récursivement ceux des sous-répertoires. Pendant le calcul (qui peut être long), nous voulons afficher les fichiers en cours de parcours. A la fin du calcul, nous voulons afficher le nombre de fichiers ainsi comptés.

Voici un code source utilisant SwingWorker qui permet de faire cela. Vous pouvez le copier-coller puis le compiler (j'ai volontairement tout écrit dans un seul fichier, pour des raisons pratiques).

```
public class SwingWorkerDemo extends JFrame {
    private JTextArea textArea;
    private JTextField textField;
    private JProgressBar progressBar;

    class MonSwingWorker extends SwingWorker<Integer,
       String> {

        public MonSwingWorker() {
            addPropertyChangeListener(new
        PropertyChangeListener() {
                public void
        propertyChange(PropertyChangeEvent evt) {
                    if("progress".equals(evt.getProperty
                            Name())) {
                        progressBar.setValue((Integer)
                            evt.getNewValue());
                    }
                }
            });
        }

        public Integer doInBackground() {
            File userDir = new
               File(System.getProperty("user.dir"));
            return getNombreDeFichiers(userDir, 0, 100);
        }

        private int getNombreDeFichiers(File dir, double
              progressStart, double progressEnd) {
            File[] files = dir.listFiles();
            int nb = 0;
            if(files.length > 0) {
                /* Le calcul de l'avancement du
                      traitement n'a que peu
                      d'importance pour l'exemple. */
                double step = (progressEnd -
                      progressStart) / files.length;

                for(File f : files) {
                    progressStart += step;

                    setProgress((int) progressStart);

                    /* Ajout d'un temps d'attente pour
                      observer les changements à
                      l'échelle "humaine". */
                    try {
                        Thread.sleep(50);
                    } catch(InterruptedException e) {
                        e.printStackTrace();
                    }

                    if(f.isDirectory()) {
                        publish("Exploration de " +
                              f.getAbsolutePath());
                        nb += getNombreDeFichiers(f,
                              progressStart,
                              progressStart + step);
                    } else {
                        publish(f.getAbsolutePath());
                        nb++;
                    }
                }
            }
            return nb;
        }

        protected void process(List<String> strings) {
            for(String s : strings)
                textArea.append(s + '\n');
        }

        protected void done() {
            try {
                /* Le traitement est terminé. */
                setProgress(100);

                textField.setText(String.valueOf(get()
                      ));
            } catch(Exception e) {
                e.printStackTrace();
            }
        }
    }

    public SwingWorkerDemo() {
        super("SwingWorkerDemo");
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        textArea = new JTextArea(12, 40);
        textArea.setEnabled(false);
        textField = new JTextField(5);
        textField.setEnabled(false);
        progressBar = new JProgressBar();
        JPanel content = new JPanel(new
               BorderLayout());
        content.add(new JScrollPane(textArea,));
        JPanel south = new JPanel(new BorderLayout());
        south.add(progressBar);
        south.add(textField, BorderLayout.EAST);
        content.add(south, BorderLayout.SOUTH);
        setContentPane(content);
        pack();
        setLocation(100, 100);
        setVisible(true);
    }

    public static void main(String... args) {
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                SwingWorkerDemo demo = new
                      SwingWorkerDemo();
                MonSwingWorker swingWorker = demo.new
                      MonSwingWorker();
                swingWorker.execute();
            }
        });
    }
}
```

Ce code utilise toutes les méthodes décrites précédemment, ce qui permet d'observer leur utilisation.

Attention : **SwingWorker** ne garantit pas que la méthode **done** sera appelée après tous les appels à **process**. Ainsi, si vous mettez dans la méthode **done** l'affichage d'une ligne supplémentaire dans le textarea, cette ligne ne se trouvera pas forcément en dernière dans le textarea.

5. Conclusion

SwingWorker permet de faciliter la transmission de messages entre un thread de calcul et **l'EDT**. Il suffit de définir la méthode qui fait le calcul en arrière-plan, de décrire les messages à transmettre à **l'EDT** et de définir les actions à effectuer dans **l'EDT** à la réception de ces messages. De plus, il est aisé de gérer une barre de progression, même si le calcul de l'avancement n'est pas toujours trivial, et est forcément erroné dans certains cas (par exemple, ici, on ne peut pas savoir l'avancement exact, puisqu'on ne connait pas le nombre total de fichiers - c'est ce que l'on cherche à calculer).

Retrouvez l'article en ligne de Romain Vimont : Lien4

Vu sur les Blogs

Eclipse 3.3 M5 disponible et Europa sur la bonne voie

Eclipse 3.3 M5 est disponible depuis la fin de la semaine dernière. On pourra noter entre autres :

- Mozilla everywhere : possibilité d'intégrer XulRunner (Lien5)
- SWT on Vista WPF : implémentation de SWT basée sur WPF, la version Win32 étant déjà disponible et s'intégrant parfaitement sur Vista
- Bundle a JRE in your RCP product : et oubliez certains problèmes de dépendances

Enfin bref, tout ça m'a rappelé que ce n'est qu'une petite partie de ce qui nous attends le 29 juin ~~juillet~~. En effet, le projet Callisto est reconduit en 2007 sous le joli nom de code Europa (Lien6). Le programme est simple : pas moins de 22 projets de la fondation Eclipse doivent être mis à jour de manière synchronisée.

Parmi les plus connus, on peut citer : WTP (Web Tool Platform), AJDT (AspectJ Development Tools), BIRT (Business Intelligence and Reporting Tools) et naturellement Eclipse (Platform, JDT, PDE).

A la fin de cette semaine (vendredi 16 février) 11 de ces projets seront disponibles en version M5 (staging +1).

Retrouvez le billet de Ioan : Lien7

La compatibilité ascendante : le plus mauvais ami de Java ?

C'est bien connu, à chacune de ses nouvelles versions, la plateforme Java prône la compatibilité ascendante : depuis toujours Java semble lui vouer un culte qui frise la déification.

Petit rappel pour ceux qui ne suivrait pas ! On pourrait brièvement décrire la compatibilité ascendante par la phrase suivante : "Le nouveau système sait faire marcher les logiciels de l'ancien, mais l'inverse n'est pas vrai".

La compatibilité constitue d'ailleurs un des 6 grands thèmes (Lien8) de **Java SE 6**

Concrètement, cela signifie qu'un programme conçu pour un JRE plus ancien continuera à fonctionner de la même manière avec un JRE plus récent, sans nécessiter de re-compilation ou de modification du code source (à quelques exceptions près plutôt rare).

Donc votre application développée pour Java 1.2 fonctionnera avec le tout dernier Java SE 6, mais l'inverse n'est logiquement pas vrai.

Toutefois, si cela a grandement contribué à l'adoption du langage, c'est également à l'origine de plusieurs frein à son l'évolution et même à la correction de bugs...

En effet, toute modification dans le langage ou dans l'API peut logiquement avoir un impact sur les programmes existants, et peut donc potentiellement nuire au bon déroulement de ces programmes, ce qui reviendrait à casser la compatibilité.

Ainsi, afin qu'une modification puisse être intégré dans le langage, elle ne doit avoir aucun impact sur les programmes existants, et ce n'est pas forcément toujours évident... un grand nombre de bugs ne peuvent donc pas être corrigé sans nuire à la compatibilité ascendante, et reste présent de version en version. Pire encore : les JVMs alternative (c'est à dire non distribué par Sun) reproduise les mêmes bugs de la JVM de Sun afin d'être totalement compatible (on parle alors de JVM "bug-for-bug" compatible).

Le problème est tout simple, et si vous suivez un peu les liens "en vrac" du blog Java, vous avez sans doute jeter un coup d'oeil à la proposition d'API de compatibilité (Lien9) sur Javalobby, qui tente d'apporter une solution à ce problème.

Sa proposition est plutôt simple (et doit surement être amélioré), et permettrait de donner la possibilité de proposer plusieurs implémentations selon une "version de compatibilité".

Ainsi une application "Java 7" pourrait se passer de bugs présent dans des JVM plus anciennes, ou encore utiliser de nouvelle fonctionnalité incompatible avec les précédentes APIs. C'est particulièrement intéressant lorsqu'on conserve une main-mise sur les machine cible, ou que l'on a des restrictions vis à vis de la JVM à utiliser.

Bien sûr cela resterait au développeur de choisir la version avec laquelle il veut rester compatible. Cela permettrait ainsi une gestion plus souple de cette compatibilité ascendante : les anciens programmes fonctionnerait toujours, mais on pourrait également choisir une "autre implémentation" pour les programmes plus récents qui n'ont pas besoin de rester compatible avec d'anciennes JVMs...

Décidément, Java 7 fait déjà beaucoup parlé de lui : est-ce un effet du passage en open-source ?

Retrouvez le billet d'adiguba: Lien10

Mise en place d'une passerelle d'authentification à l'aide de authpf

1. Présentation

Une passerelle d'authentification par rapport à une passerelle normale demande à ce que l'utilisateur s'identifie afin que le trafic généré par ce dernier puisse la traverser. Pour cela chaque utilisateur se voit attribuer un shell différent de ceux que nous connaissons déjà : authpf, et lors d'une connexion SSH réussie à la passerelle, ce shell chargera dynamiquement les règles correspondant à cet usager.

Ces règles peuvent être globales ou bien spécifiques, c'est-à-dire propres à chacun. Il est ensuite possible d'appliquer sans restriction tout type de règles proposé par Packet Filter : passer au travers du filtre, redirection, traduction d'adresse (NAT), etc.

Les connexions et déconnections sont enregistrées par authpf via syslog, fournissant ainsi à l'administrateur l'adresse IP de la machine utilisée, le login de l'utilisateur, l'heure de l'établissement de la session, l'heure à laquelle la session s'est terminée et sa durée. Vous trouverez ici un exemple de parser écrit en Perl.

Quelques utilisations courantes :

- Exiger une authentification de la part des utilisateurs avant de leur permettre d'utiliser ou d'avoir accès aux ressources réseaux (Intranet comme Internet) :
  - Rendre possible le travail à domicile
  - Accès à un réseau sans fil
- Adapter le dispositif de filtrage à l'utilisateur connecté :
  - Donner des accès supplémentaires à des administrateurs
  - Etre plus permissif envers les utilisateurs connus
  - Rediriger vers la machine attribuée à un utilisateur

2.Attribution du shell authpf

Tout d'abord il est nécessaire que authpf soit reconnu comme étant un shell par le système, ce qui n'est pas le cas par défaut. Pour cela exécutez la commande :

```
grep '^/usr/sbin/authpf$' /etc/shells || echo
'/usr/sbin/authpf' >> /etc/shells
```

2.1. Attribution individuelle : changement de shell

Dans le cas d'utilisateurs déjà présents sur le système, il faut modifier manuellement le shell de chacun d'eux. Voici une liste (non exhaustive) des commandes permettant d'effectuer cette opération :

Plusieurs méthodes sont possibles :

- A l'aide de chsh. Exemple d'utilisation :

```
chsh -s /usr/sbin/authpf login_utilisateur
```

- Un équivalent très proche, basé sur la commande **pw** :

```
pw usermod login_utilisateur -s /usr/sbin/authpf
```

- Editez tous les comptes présents sur la machine à l'aide de **vipw** pour modifier les shells des utilisateurs concernés en /usr/sbin/authpf (le shell étant le dernier champ).

A la création d'utilisateurs avec la commande **adduser** vous pourrez spécifier authpf comme shell.

2.2. Attribution groupée : utilisation des classes de session

Méthode particulièrement adaptée si les utilisateurs devant utiliser ce mécanisme d'authentification ne sont pas encore créés.

Nous allons créer une nouvelle classe pour les utilisateurs concernés dans notre fichier /etc/login.conf telle que :

```
authpf:\
    :shell=/usr/sbin/authpf:\
    :tc=default:
```

Il vous est possible d'ajouter de nombreuses limitations aux utilisateurs associés à une classe. Pour vous en rendre compte consultez la page man de /etc/login.conf (Lien11).

Après avoir effectuer des changements dans ce fichier il est impératif d'en regénérer la base afin de les prendre en considération. Utiliser la commande **cap_mkdb** à cette fin :

```
cap_mkdb /etc/login.conf
```

Vous pourrez alors utiliser cette nouvelle classe lors de la création de nouveaux utilisateurs avec adduser.

En revanche, si vos utilisateurs existent déjà, vous pouvez utiliser la commande pw pour les changer de classe. Voici un exemple où l'utilisateur julp se voit attribuer cette classe :

```
pw usermod julp -L authpf
```

3. Configuration de Packet Filter

3.1. Pré-requis

Le répertoire accueillant les fichiers de configuration de authpf n'existant pas par défaut, il nous faut donc le créer :

```
mkdir /etc/authpf/
```

Authpf requiert également un autre répertoire, qui lui héberge des fichiers temporaires dont les noms sont les adresses IP des machines distantes où une connexion SSH est ouverte pour l'exécuter. Sa particularité c'est que le groupe du même nom doit avoir tous les droits sur celui-ci :

```
mkdir /var/authpf/
chmod 0770 /var/authpf
chgrp authpf /var/authpf
```

Dernière spécificité, et non des moindres, ce shell utilise le système de fichier nommé fdescfs, qui permet d'accéder à un fichier (déjà ouvert) à partir de son descripteur. Afin de le prendre en charge, nous ajoutons dans /etc/fstab, la ligne suivante :

```
# Device    Mountpoint  FStype   Options   Dump    Pass#
fdescfs     /dev/fd     fdescfs  rw        0       0
```

3.2. Configuration

Authpf refusera systématiquement toute connexion en l'absence du fichier /etc/authpf/authpf.conf, c'est pourquoi il faut absolument le créer même vide. Il permet de spécifier deux options de configuration :

- **anchor** : définit le nom de l'ancre. Sa valeur par défaut est authpf.
- **table** : permet de spécifier le nom de la table contenant l'ensemble des adresses IP d'où les utilisateurs sont actuellement connectés. Sa valeur par défaut est authpf_users.

Voilà comment il sera défini pour la suite de cet article :

```
/etc/authpf/authpf.conf :
anchor=authpf
table=utilisateurs_authpf
```

3.3. Règles propres à chaque utilisateur

Authpf présente également l'intérêt de pouvoir charger des règles de filtrage spécifiques à chaque utilisateur. Pour cela, il suffit de les écrire dans un fichier de la forme /etc/authpf/users/$USER/authpf.rules où $USER correspond au login de l'utilisateur en question. Pas d'inquiétude, vous n'aurez pas réécrire les règles pour chaque utilisateur puisque authpf propose d'utiliser des règles par défaut définies dans /etc/authpf/authpf.rules si ce dernier n'existait pas.

Il paraît dès lors évident qu'au moins un de ces deux fichiers doit toujours être disponible quelque soit l'utilisateur qui se connectera. En effet, le shell mettra immédiatement fin à la connexion de l'utilisateur si aucune règle (par défaut ou non) ne peut lui être appliquée. Par conséquent, il est préférable de définir dans /etc/authpf/authpf.rules les règles s'appliquant aux utilisateurs lambda ou de le laisser vide si celles-ci sont directement définies parmi vos règles principales. Nous y venons justement dans la prochaine partie.

Deux macros sont automatiquement disponibles pour la création de vos règles dans les fichiers cités ci-dessus, à savoir :

- **$user_ip** : l'adresse IP de la machine depuis laquelle l'utilisateur a établi la connexion vers la passerelle
- **$user_id** : l'identifiant (login) de l'utilisateur en question

Vous trouverez plus bas dans l'exemple intitulé Autoriser l'accès au réseau sans fil une utilisation de celles-ci.

3.4. Règles principales

Voyons comment sont gérées les règles dans le fichier principal de configuration de Packet Filter. Ci-dessous une ébauche générique complète qui vous servira de base. Vous trouverez les explications correspondantes après celui-ci :

```
##### Macros #####
interface = "de0"

# Table contenant la liste des adresses IP des
utilisateurs actuellement connectés
table <utilisateurs_authpf> persist

# Ancre pour la traduction d'adresses
# (une adresse IP publique pour l'ensemble du réseau
interne)
nat-anchor "authpf/*"
# Ancre pour les redirections
rdr-anchor "authpf/*"
# Ancre pour la traduction d'adresses
# (chaque machine possède une adresse IP publique et
privée)
binat-anchor "authpf/*"

##### Filtrage #####
# Règle par défaut : tout bloquer
block drop all

# SSH accepté
pass in quick on $interface inet proto tcp from any \
    to $interface port ssh flags S/SA keep state

# Ancre pour le filtrage
anchor "authpf/*"
```

La déclaration de la table dont vous avez donné le nom par la directive table dans /etc/authpf/authpf.conf n'est nécessaire que si vous envisagez de l'utiliser. Celle-ci contiendra la liste des adresses IP des machines depuis lesquelles une personne s'est authentifiée à la passerelle. La présence du mot-clé persist est obligatoire pour maintenir son existence en mémoire même si celle-ci venait à être vide.

Il existe quatre types d'ancres, comme montré ci-dessus, dont chacune dépend de la nature des règles qui seront dynamiquement chargées. Cette dernière est indiquée par son préfixe. Toutes ne sont pas requises, vous ne pouvez faire figurer que celles dont vous avez besoin ainsi dans le cas où vous ne faites aucune traduction d'adresse vous pouvez omettre les lignes commençant par nat-anchor et binat-anchor.

Enfin, on notera la présence impérative d'une règle qui autorise le trafic SSH en vue de permettre l'authentification de nos différents utilisateurs.

4. Administration et gestion des utilisateurs

4.1. Gestion des utilisateurs

L'administrateur peut décider de bannir (bloquer) un utilisateur, pour cela il lui suffit de créer le fichier /etc/authpf/banned/$USER, où $USER désigne le login de cet utilisateur. Ce fichier peut avoir pour contenu la raison de cette interdiction et ce message sera affiché lors de la connexion de celui-ci par SSH. En guise d'exemple, nous souhaitons interdire l'utilisateur julp :

```
echo "Interdit de réseau pour utilisation de réseaux
pear to pear." > /etc/authpf/banned/julp
```

A l'inverse le fichier /etc/authpf/autpf.allow indique les utilisateurs autorisés à avoir recours à authpf en y spécifiant leurs logins. Toutefois, si ce dernier n'est pas présent sur votre système ou bien s'il a pour contenu * alors tout le monde est autorisé à s'y connecter.

Le fichier d'interdiction est toujours cherché avant celui d'autorisation. En clair, l'interdiction prévaut sur l'autorisation. De plus, si authpf n'est pas capable de déterminer l'un ou l'autre alors il met fin à la connexion SSH.

Il est dès lors possible de déterminer deux approches :

- politique permissive : placer * dans /etc/authpf/authpf.allow et bloquer les utilisateurs au cas par cas
- politique restrictive : ne lister que les personnes autorisées dans /etc/authpf/authpf.allow

4.2. Lister les utilisateurs actuellement connectés

Le moyen le plus commode pour savoir qui est en train d'utiliser la passerelle et depuis où reste ps. En effet, lorsque la connexion est établie avec succès, authpf modifie le nom de son processus pour y intégrer le login de l'utilisation ainsi que l'adresse IP de la machine cliente. En voici une démonstration :

```
# ps ax | grep authpf
 1123  p0  Ss+    0:00,05 -authpf:
stéphane@192.168.100.1 (authpf)
```

Il est possible de mettre fin à la connexion de stéphane établie depuis la machine d'adresse 192.168.100.1 en envoyant un signal TERM (15) au processus associé à celle-ci. Dans de tel cas, Authpf se chargera de la suppression des connexions dont l'état a été mémorisé (option keep state et similaires) et des règles spécifiques à cet utilisateur s'il y a lieu, de nettoyer la table.

```
kill -TERM 1123
# ou encore
kill -15 1123
```

Toutefois, pour des questions de praticité, j'ai mis au point un petit programme qui vous permettra de lister et de mettre fin à une session. Celui-ci ne possède aucune dépendance particulière. Les sources sont à votre disposition ici (Lien12).

4.3. Affichage de messages de bienvenue et d'erreur

Il existe plusieurs méthodes pour afficher un message particulier à l'ensemble des utilisateurs recourant à authpf. Voici la liste des fichiers de message reconnus par authpf :

- /etc/authpf/authpf.message est un fichier dont le contenu est affiché, lorsqu'il existe, après authentification d'un utilisateur (comprenez par là que celle-ci est fructueuse).
- /etc/authpf/authpf.problem en sa présence, authpf imprimera son contenu sur la sortie standard lorsqu'il rencontrera une erreur en plus d'être écrite dans le journal système via syslog. Il peut être utilisé afin que le service informatique soit informé de ce problème technique par les usagers.

Voyons un exemple :

```
echo -n > /etc/authpf/authpf.message

cat > /etc/authpf/authpf.problem
Il semblerait que le système rencontre quelques
difficultés techniques.

Reportez ce problème en téléphonant au 0X-XX-XX-XX-XX
ou en envoyant un
email à administrateur@domaine.fr afin que nous
puissions le résoudre
au plus vite.
^D

cat > /etc/authpf/authpf.motd
Vous pouvez maintenant accéder au réseau. Cette
connexion n'est pas anonyme
et vous devenez à cet instant responsables du trafic
généré à partir de votre
machine.
^D
```

Ici /etc/authpf/authpf.motd est un fichier qui servira à cultiver l'obscurantisme, c'est-à-dire à ne pas dévoiler une quelconque information au sujet du système d'exploitation (typiquement ses nom et version). Par contre, son recours requiert l'appel à une classe de session et nous complèterons celle que nous avons défini plus haut, dans la partie Attribution groupée : utilisation des classes de session, dans /etc/login.conf tel que :

```
authpf:\
        :shell=/usr/sbin/authpf:\
        :welcome=/etc/authpf/authpf.motd:\
        :tc=default:
```

Regénérons la base de données :

```
cap_mkdb /etc/login.conf
```

/etc/authpf/authpf.motd sera affiché lorsque tout va bien, c'est pourquoi nous créons /etc/authpf/authpf.message vide.

4.4. Journalisation des évènements

La question de journalisation a été abordée précédemment lors de la présentation de authpf et celle-ci est inactive par défaut. Nous allons modifier /etc/syslog.conf pour la prendre en charge :

```
!authpf
*.*                     /var/log/authpf.log
```

Puis nous procédons ensuite à la création du fichier de log qui sera dédié à authpf :

```
touch /var/log/authpf.log
chmod 0600 /var/log/authpf.log
```

Nous terminons par le redémarrage du démon syslog afin que les modifications apportées soient prises en compte dès maintenant :

```
/etc/rc.d/syslogd restart
# ou
pkill -1 syslogd
```

4.5. La sécurité autour de SSH

4.5.1. Paramétrage de SSH

Il peut s'avérer nécessaire de reconfigurer le service SSH au niveau de la sécurité puisque SSH est la clé d'accès au réseau pour vos utilisateurs. Voici la liste des options qui nous intéressent :

- Protocol 2 : seule la version 2 du protocole est acceptée et gérée par sshd. Préférable pour des raisons de sécurité.
- AllowTcpForwarding no : il est recommandé de refuser les redirections TCP car celles-ci peuvent entraîner le blocage des paquets par PF.
- ClientAliveInterval 15 : fréquence, en secondes, de vérification de la validité de la connexion du client. Option ne s'appliquant qu'à la version 2 du protocole - valeur par défaut 0, inactive.
  Combinée à la directive ClientAliveCountMax (présentée ci-dessous), elle permet de terminer les connexions inactivent ou qui font l'objet d'une tentative de vol par usurpation (spoofing).
- ClientAliveCountMax 3 : nombre maximum de vérifications infructueuses acceptées par sshd avant de

mettre fin à la session.

4.5.2. Se prémunir contre les attaques par dictionnaire

Les systèmes équipés de la version 3.7 de Packet Filter ou supérieure (ce qui correspond pour FreeBSD à une version supérieure ou égale à 6.0) peuvent bloquer les tentatives d'attaque par dictionnaire à l'aide des règles suivantes :

```
##### Macros #####
ext_if = "votre interface sur l'extérieur"

##### Déclaration des tables #####
table <ssh-bruteforce> persist [file "/etc/ssh-
bruteforce"] # La partie file est optionnelle

##### Règles #####
# [...]

# SSH
block in quick from <ssh-bruteforce>
pass in quick on $ext_if inet proto tcp from any \
    to $ext_if port ssh flags S/SA keep state (max-src-
conn-rate 4/30, overload <ssh-bruteforce> flush global)

# [...]
```

Ainsi, lorsque quelqu'un tentera 4 connexions en moins de 30 secondes (option max-src-conn-rate 4/30) vers ssh, son adresse IP sera ajoutée à la table ssh-bruteforce (option overload) et mettra fin à toutes ses connexions (option flush global).

5. Après la théorie : la pratique par l'exemple

Les exemples suivants sont volontairement simplifiés pour mettre en lumière des applications courantes de authpf ainsi que pour montrer l'intégration des règles relatives à authpf par rapport à celles qui sont statiques.

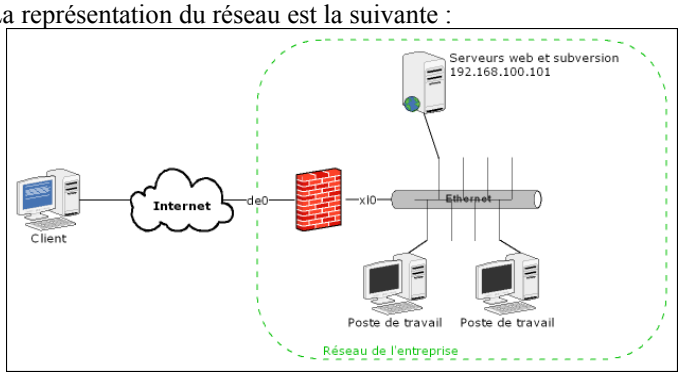
Les pare-feu dont les configurations sont données en exemple sont marqués sur les schémas ci-dessous sous les traits d'un mur en brique.

5.1. Permettre aux utilisateurs de travailler depuis chez eux

Le directeur d'une petite entreprise spécialisée dans la création de site web souhaite, suite aux demandes répétées de ses salariés, leur permettre de travailler depuis chez eux.

Cet accès se limitera au bastion faisant office de serveur Web et de serveur de gestion de configuration (subversion), leur offrant ainsi la possibilité de récupérer, mettre à jour les sources et visualiser directement le résultat de ces changements (aucune installation locale particulière n'est ainsi requise).

La représentation du réseau est la suivante :



Le fichier /etc/authpf/authpf.rules sera créé vide pour que l'ensemble des règles figure dans un seul et unique fichier. Voici comment il sera écrit :

```
##### Macros #####
ext_if = "de0"
int_if = "xl0"
tcpflags = "flags S/SFRA"
http_server = "192.168.100.101"
nat_ports = "port 50000:60000"

##### Options #####
set skip on lo0       # L'interface de loopback est
ignorée
set block-policy drop # Les paquets sont
silencieusement bloqués

##### Déclaration des tables #####
table <utilisateurs_authpf> persist

##### Normalisation des paquets #####
scrub in

##### Traduction d'adresse #####
nat on $ext_if from !($ext_if) \
    to any -> ($ext_if) $nat_ports

##### Redirections #####
rdr on $ext_if inet proto tcp from
<utilisateurs_authpf> \
    to ($ext_if) port http -> $http_server port http

##### Règles #####
block all

#SSH autorisé
pass in quick on $ext_if inet proto tcp from any \
    to ($ext_if) port ssh $tcpflags keep state

#HTTP autorisé de et vers les utilisateurs authentifiés
pass in quick on $ext_if inet proto tcp from
<utilisateurs_authpf> \
    to $http_server port http $tcpflags keep state
pass out quick on $int_if inet proto tcp from
<utilisateurs_authpf> \
    to $http_server port http $tcpflags keep state

# Trafic TCP traduit autorisé à sortir
pass in quick on $ext_if inet proto tcp from
$int_if:network \
    to any $tcpflags keep state
pass out quick on $ext_if inet proto tcp from $ext_if
$nat_ports \
    to any $tcpflags keep state

# Ancre permettant à authpf de charger dynamiquement
des règles propres à chaque utilisateur
anchor "authpf/*" in on $ext_if
```

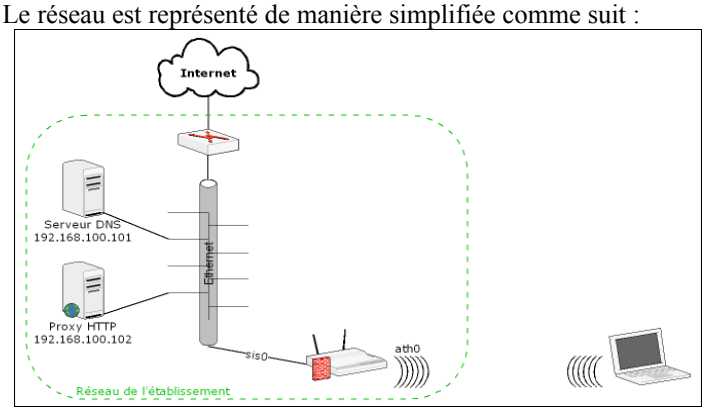
5.2. Autoriser l'accès au réseau sans fil

Un établissement scolaire offre un accès sans-fil pour permettre aux élèves comme aux enseignants de consulter des sites Internet. Ce trafic est contrôlé en ayant recours à des serveurs de mandatement internes pour les protocoles HTTP et DNS et doit donc obligatoirement passer par eux. De plus, ces utilisateurs devront montrer patte blanche avant d'accéder aux ressources réseau mises à leur disposition.

Les administrateurs systèmes et réseaux, qui seront également amenés à recourir au sans-fil pour travailler, souhaitent disposer d'un accès administratif (SSH) vers les machines qui assurent des

services mandataires.

Le réseau est représenté de manière simplifiée comme suit :



Les règles générales sont, par conséquent, définies telles que :

```
##### Macros #####
ext_if = "ath0"
int_if = "sis0"
tcpflags = "flags S/SFRA"
dns_server = "192.168.100.101"
http_proxy = "192.168.100.102"

##### Options #####
set skip on lo0
set block-policy drop

##### Déclaration des tables #####
table <utilisateurs_authpf> persist

##### Normalisation des paquets #####
scrub in

##### Règles #####
block all

# DHCP autorisé pour des raisons de commodité
pass in quick on $ext_if inet proto tcp from any \
    to $ext_if port bootpc $tcpflags keep state

# SSH autorisé
pass in quick on $ext_if inet proto tcp from
$ext_if:network \
    to $ext_if port ssh $tcpflags keep state

# HTTP autorisé vers le mandataire pour les
utilisateurs authentifiés
pass in quick on $ext_if inet proto tcp from
<utilisateurs_authpf> \
    to $http_proxy port { http https } $tcpflags keep
state

# DNS autorisé vers notre serveur DNS (cache et
forwarding) pour les utilisateurs authentifiés
pass in quick on $ext_if inet proto udp from
<utilisateurs_authpf> \
    to $dns_server port domain keep state

# Ancre permettant à authpf de charger dynamiquement
des règles propres à chaque utilisateur
anchor "authpf/*" in on $ext_if
```

Enfin, nous attribuons des règles spécifiques à l'administrateur dupont.j lui permettant d'avoir un accès SSH vers le serveur DNS et le mandataire HTTP depuis son portable équipé de la technologie Wifi. Pour cela nous créons /etc/authpf/users/dupont.j/authpf.rules pour lui donner cet accès :

```
##### Macros #####
ext_if = "ath0"
tcpflags = "flags S/SFRA"
dns_server = "192.168.100.101"
http_proxy = "192.168.100.102"

# SSH permis vers le serveur DNS et le proxy
pass in quick on $ext_if inet proto tcp from $user_ip \
    to { $dns_server $http_proxy } port ssh $tcpflags
keep state
```

Veillez à recopier ou remplacer les macros que vous souhaitez utiliser dans de tels fichiers de règles car celles-ci ne sont pas héritées du fichier de règles principales.

6. Epilogue

Authpf est un programme fiable basé sur le mécanisme appelé ancre de Packet Filter qui permet de charger dynamiquement des règles. Cette fiabilité est en grande partie assurée par le protocole SSH dont il ne faut pas, par conséquent, négliger la configuration. Sa capacité à journaliser les évènements de démarrage et fin de session raviront sans aucun doute les administrateurs.

Toutefois, sous forme d'interpréteur de commande, authpf restreint l'utilisation du compte utilisateur à cet usage : il ne sera en effet impossible d'utiliser SSH pour exécuter des commandes.

Lire l'article de julp en ligne : Lien13

Vu sur les Blogs

Les solutions Linux 2007 c'est fini !

Comme vous le savez le salon Solutions Linux 2007 s'est tenu du 30 janvier au 1er Février au CNIT de la Défense.

Ce salon a été l'occasion pour nous de faire plusieurs choses, la première de nos tâches etait de nous présenter à ceux d'entre-vous qui ne nous connaissaient pas encore, nous avions comme l'année passée un stand pour cela.

Nous avons également filmé les conférences gratuites en partenariat avec Tarsus, l'organisateur, et Novell. Vous pourrez retrouver toutes ces videos dans quelques jours sur l'espace Developpez.tv de notre site.

Retrouvez le billet de Gaël Donat : Lien14

Retrouvez les vidéos du salon Solutions Linux : Lien15

Génération de documents PDF en utilisant la librairie FPDF

1. Présentation

1.1. Le projet

L'article présenté utilise comme base la classe **FPDF**.

FPDF est une classe PHP permettant de générer des documents PDF en pur PHP, c'est-à-dire sans utiliser la librairie PDFlib. Le F de **FPDF** signifie Free : vous êtes libres de l'utiliser et de la modifier comme vous le souhaitez. Plusieurs développeurs ont ajouté des classes pour créer à chaque fois de nouvelles choses. La classe de base est **FPDF** et dès que quelqu'un veut ajouter un script, il crée une classe qui en hérite.

Voulant moi-même utiliser plusieurs classes déjà existantes, j'ai copié leurs contenus dans la nouvelle classe **phpToPDF** qui hérite de **FPDF**.

FPDF a d'autres avantages : des méthodes de plus haut niveau.

Les principales fonctionnalités de **FPDF**
- Choix des unités, du format des pages et des marges;
- Gestion des en-têtes et pieds de page;
- Saut de page automatique;
- Saut de ligne automatique et justification;
- Images (JPEG et PNG);
- Couleurs;
- Liens;
- Support des polices TrueType et Type1;
- Compression des pages.

FPDF ne nécessite aucune extension (à part zlib pour activer la compression) et fonctionne avec **PHP 4 et PHP 5.**

1.2. Installation

Il suffit de télécharger et de mettre dans le répertoire racine de son site :
- Les sources PHP fpdf.php et phpToPDF.php;
- Le répertoire "font/" qui contient les fonts.

Télécharger phpToPDF.zip : Lien16

Quand vous avez installé (copié) les scripts PHP et le répertoire "font" sur votre serveur, vous êtes prêts à générer des documents PDF à partir d'un script PHP.

1.3. La documentation

Les méthodes de base (issues de la classe FPDF) ne sont pas toutes détaillées ici. Vous pouvez voir les descriptions et paramètres sur le site officiel (Lien17)

2. Premiers pas

Il faut savoir que les méthodes utilisées écrivent dans une page du document PDF généré. Avant d'utiliser certaines méthodes, il faut se placer dans la page avec la méthode SetXY(x,y) car on ne peut pas passer en paramètre la position désirée. (cf. l'exemple avec la méthode Cell)

Il est obligatoire de mettre la ligne SetFont sinon, la génération ne fonctionne pas...

2.1. Texte seulement

Il y a plusieurs façons d'écrire un texte dans une page.

2.1.1. avec la méthode Text

Description
Imprime une chaîne de caractères. L'origine est à gauche du premier caractère, sur la ligne de base. Cette méthode permet de positionner précisément une chaîne dans la page, mais il est généralement plus simple d'utiliser Cell(), MultiCell() ou Write() qui sont les méthodes standards pour imprimer du texte.

```
include("phpToPDF.php");

$PDF = new phpToPDF();
$PDF->AddPage();
$PDF->SetFont("Arial","B",16);
$PDF->Text(40,10,"Uniquement un texte");
$PDF->Output();
```

Commentaires
- AddPage: ajoute une page dans le document;
- SetFont: détermine la font utilisée (B pour Bold);
- Text(float x, float y, string txt): Dans l'exemple ci-dessus, Text écrit "Uniquement un texte" en position (40, 10);
- Output: permet d'afficher le document généré dans le navigateur.

2.1.2. avec la méthode Write

Description
Cette méthode imprime du texte à partir de la position courante. Lorsque la marge droite est atteinte (ou que le caractère \n est rencontré), un saut de ligne est effectué et le texte continue à partir de la marge gauche. Au retour de la méthode, la position courante est située juste à la fin du texte. Il est possible de mettre un lien sur le texte.

```
include("phpToPDF.php");

$PDF=new phpToPDF();
$PDF->AddPage();
$PDF->SetFont('Arial','B',16);
$PDF->Write(10, "Ceci est un texte multilignes \nEt
voici la deuxième ligne");
$PDF->Output();
```

Commentaires

- **Write(float h, string txt [, mixed link])** Dans l'exemple ci-dessus, **Write** écrit le texte "Ceci est un texte multilignes \nEt voici la deuxième ligne" avec un saut de ligne de 10 mm.

2.1.3. avec la méthode Cell

Description

Imprime une cellule (zone rectangulaire) avec éventuellement des bords, un fond et une chaîne de caractères. Le coin supérieur gauche de la cellule correspond à la position courante. Le texte peut être aligné ou centré. Après l'appel, soit la position courante se déplace à droite, soit un retour à la ligne est effectué. Il est possible de mettre un lien sur le texte.

```
include("phpToPDF.php");

$PDF = new phpToPDF();
$PDF->AddPage();

//Sélection de la police
$PDF->SetFont('Arial','B',16);

//Décalage de 8 cm à droite
$PDF->Cell(80);

//Texte centré dans une cellule 20*10 mm encadrée et
retour à la ligne
$PDF->Cell(20,10,'Titre',1,1,'C');
$PDF->Output();
```

Commentaires

- **Cell(80);** écrit une cellule vide sans bord de 80 mm de large à partir de l'endroit où l'on se trouve, c'est-à-dire par défaut, en position (margeLeft, margeTop). Les marges ont la valeur 10 mm par défaut, pour les changer, utiliser SetMargins(); L'appel setXY(10, 90); aurait été similaire;
- **Cell(float w [, float h [, string txt [, mixed border [, int ln [, string align [, int fill [, mixed link]]]]]])** Dans l'exemple ci-dessus, Cell écrit une cellule de taille (20,10), contenant le texte 'Titre', avec un bord, retour à la ligne et centré.

2.1.4. avec la méthode MultiCell

Description

Cette méthode permet d'imprimer du texte avec des retours à la ligne. Ceux-ci peuvent être automatiques (dès que le texte atteint le bord droit de la cellule) ou explicites (via le caractère \n). Autant de cellules que nécessaire sont imprimées, les unes en dessous des autres.
Le texte peut être aligné, centré ou justifié. Le bloc de cellules peut être encadré et le fond coloré.

```
include("phpToPDF.php");

$PDF = new phpToPDF();
$PDF->AddPage();

//Sélection de la police
$PDF->SetFont('Arial','B',16);

$PDF->MultiCell(0, 10, "Ceci est un texte multilignes
centré avec un bord\nEt voici la deuxième ligne", 1,
"C", 0);
$PDF->Output();
```

Commentaires

- **MultiCell(float w, float h, string txt [, mixed border [, string align [, int fill]]]);** Dans l'exemple ci-dessus, MultiCell écrit une cellule de taille (0, 10), contenant le texte entre guillemets, avec un bord, centré et sans remplissage de la cellule.

2.2. Image seulement

Description

C'est la méthode Image(string file, float x, float y [, float w [, float h [, string type [, mixed link]]]]) qui est utilisée. Elle place une image dans la page. Le coin supérieur gauche doit être spécifié.

Les dimensions peuvent être indiquées de plusieurs manières :

- largeur et hauteur explicites (exprimées dans l'unité utilisateur);
- Une dimension explicite, l'autre étant calculée automatiquement afin de respecter les proportions de l'image originale;
- Aucune dimension explicite, auquel cas l'image est dimensionnée en 72 dpi.

Les formats supportés sont le JPEG et le PNG.

```
include("phpToPDF.php");

$PDF=new phpToPDF();
$PDF->AddPage();
$PDF->SetFont('Arial','B',16);
$PDF->Image("./images/kitlogo.jpg", 50, 100);
$PDF->Output();
```

Commentaires

- Image(string file, float x, float y [, float w [, float h [, string type [, mixed link]]]]) Dans l'exemple ci-dessus, Image met l'image "./images/kitlogo.jpg" en position (50, 100)

2.3. Sommaire et numéros de page

Description

Il est aussi possible de numéroter ses pages et de générer automatiquement un sommaire. Il suffit de spécifier la page à partir de laquelle vous voulez commencer la numérotation et celle à partir de laquelle vous voulez arrêter la numérotation. Ensuite, pour chaque item du sommaire, vous devez ajouter son nom. Pour finir, il faut ajouter le sommaire sur la page désirée.

```
include("phpToPDF.php");

$PDF=new phpToPDF();
$PDF->SetFont('Times','',12);
$PDF->AddPage();
$PDF->Cell(0,5,'Page de garde',0,1,'C');
```

```
$PDF->AddPage();

// A partir de cette page, la numérotation commence...
$PDF->startPageNums();
$PDF->Cell(0,5,'TOC1',0,1,'L');

// On ajoute un item au sommaire
$PDF->TOC_Entry('TOC1', 0);
$PDF->Cell(0,5,'TOC1.1',0,1,'L');

// On ajoute un item au sommaire
$PDF->TOC_Entry('TOC1.1', 1);
$PDF->AddPage();
$PDF->Cell(0,5,'TOC2',0,1,'L');

// On ajoute un item au sommaire
$PDF->TOC_Entry('TOC2', 0);
$PDF->AddPage();
for($i=3;$i<=80;$i++){
        $PDF->Cell(0,5,'TOC'.$i,0,1,'L');

        // On ajoute un item au sommaire
        $PDF->TOC_Entry('TOC'.$i, 0);
}

// On arrête ici la numérotation
$PDF->stopPageNums();
$PDF->AddPage();
$PDF->Cell(0,5,'Page non numérotée',0,1,'L');

//Génère et insère le sommaire en page 2
$PDF->insertTOC(2);
$PDF->Output();
```

Commentaires

- **startPageNums()** Cette méthode commence la numérotation des pages à partir de la page courante;
- **TOC_Entry('titre', 0);** Cette méthode ajoute l'entrée 'titre' au sommaire;
- **stopPageNums()** Cette méthode termine la numérotation sur la page courante;
- **insertTOC(2)** Cette méthode génère le sommaire en page 2 du document.

2.4 Enregistrer dans un document

Description

Il est intéressant d'afficher directement le document généré avec la méthode Output() mais il est d'autant plus intéressant d'enregistrer le document généré sur le serveur et de l'afficher à n'importe quel moment et n'importe où dans son site. Voici la procédure...

```
include("phpToPDF.php");

$PDF=new phpToPDF();
$PDF->SetFont('Times','',12);
$PDF->AddPage();
// on écrit ce que l'on veut dans le document PDF...

// enregistre le document test.PDF dans le répertoire
local du serveur.
$PDF->Output("test.PDF", "F");

// affiche le document test.PDF dans une iframe.
echo '
        <iframe src="test.PDF" width="100%"
height="100%">
        [Your browser does <em>not</em> support
<code>iframe</code>,
        or has been configured not to display inline
frames.
        You can access <a href="./test.PDF">the
document</a>
        via a link though.]</iframe>
's;
```

Commentaires

- **Output("test.PDF, "F")** Cette méthode enregistre le document généré dans le document ./test.PDF du serveur;
- **le bloc echo** '...' Ce bloc permet d'afficher (si possible) le document ./test.PDF dans une iframe.

3. Utilisation avancée

3.1. Insérer un graphique

Nous allons voir ici comment dessiner des droites en couleur dans un repère orthonormé avec un titre, une légende, l'affichage des abscisses et des ordonnées.

1) D'abord le code minimum pour débuter son script PHP...

```
include("phpToPDF.php");

$PDF=new phpToPDF();
$PDF->AddPage();
$PDF->SetFont('Arial','B',16);
```

2) Ensuite, la création des droites que nous voulons afficher...

Elles ne sont pas définies mathématiquement (y=ax+b) mais elles sont définies par deux ordonnées, y1 et y2. Ce graphique peut donc nous montrer une progression avec ou sans intervalle.

```
$droite1 = array(0, 100, array(255,0,0), "droite 1");
$droite2 = array(50, 25, array(0,255,0), "droite 2");
$droite3 = array(12, 45, array(0,0,255), "");
$droites = array($droite1, $droite2, $droite3);
```

Commentaires

- array(0, 100, ...); 0 et 100 sont les ordonnées y1 et y2;
- array(..., array(255,0,0),...); array(255,0,0) définit la couleur de la droite;
- array(..."droite 1"); "droite 1" définit le nom de la droite dans la légende. Si cet argument est "", il n'y aura pas de légende pour cette droite;
- $droites = array($droite1, $droite2, $droite3); C'est cette variable qui sera envoyée à la méthode setRepere.

3) Enfin, l'appel de la méthode setRepere pour générer le graphique et l'affichage du document PDF.

```
$PDF->setRepere("Titre du graphique", 30, 80, 100, 60,
array("Evolution du PIB de la Creuse en 1956"),
array(0, 100, 5), $droites);
$PDF->Output();
```

- Commentaires sur les arguments de la méthode setRepere($titre, $posX, $posY, $sizeX, $sizeY, $datasX, $datasY, $droites)
- $titre C'est le titre du graphique (centré en haut du graphique);
- $posX et $posY Ce sont les coordonnées du coin en haut à gauche du graphique;

- $sizeX et $sizeY Ce sont les dimensions du graphique;
- $datasX C'est un tableau à un ou deux éléments. S'il n'y a qu'un élément, c'est une légende de l'axe des abscisses, sinon, le premier élément correspond au départ de la droite et le deuxième correspond à la fin de la droite (ex: array("année N-1", "année N"));
- $datasY C'est un tableau dont les deux premiers éléments sont les min et max de l'ordonnée. Le troisième argument est le nombre d'intervalles entre les min et max;
- $droites C'est le tableau qui contient les droites. (cf. 2)

3.2. Insérer un tableau

Nous allons voir ici comment définir un tableau composé d'un header et d'un contenu.

3.2.1. Un tableau simple

1) D'abord le code minimum pour débuter son script PHP...

```
include("phpToPDF.php");

$PDF=new phpToPDF();
$PDF->AddPage();
$PDF->SetFont('Arial','B',16);
```

2) Ensuite, la création des variables utilisées pour la génération du tableau.

```
// Définition des propriétés du tableau.
$proprietesTableau = array(
	'TB_ALIGN' => 'L',
	'L_MARGIN' => 15,
	'BRD_COLOR' => array(0,92,177),
	'BRD_SIZE' => '0.3',);

// Définition des propriétés du header du tableau.
$proprieteHeader = array(
	'T_COLOR' => array(150,10,10),
	'T_SIZE' => 12,
	'T_FONT' => 'Arial',
	'T_ALIGN' => 'C',
	'V_ALIGN' => 'T',
	'T_TYPE' => 'B',
	'LN_SIZE' => 7,
	'BG_COLOR_COL0' => array(170, 240, 230),
	'BG_COLOR' => array(170, 240, 230),
	'BRD_COLOR' => array(0,92,177),
	'BRD_SIZE' => 0.2,
	'BRD_TYPE' => '1',
	'BRD_TYPE_NEW_PAGE' => '',);

// Contenu du header du tableau.
$contenuHeader = array(
	50, 50, 50,
	"Titre de la première colonne", "année N-1",
"année N",);

// Définition des propriétés du reste du contenu du
tableau.
$proprieteContenu = array(
	'T_COLOR' => array(0,0,0),
	'T_SIZE' => 10,
	'T_FONT' => 'Arial',
	'T_ALIGN_COL0' => 'L',
	'T_ALIGN' => 'R',
	'V_ALIGN' => 'M',
	'T_TYPE' => '',
	'LN_SIZE' => 6,
	'BG_COLOR_COL0' => array(245, 245, 150),
	'BG_COLOR' => array(255,255,255),
	'BRD_COLOR' => array(0,92,177),
	'BRD_SIZE' => 0.1,
	'BRD_TYPE' => '1',
	'BRD_TYPE_NEW_PAGE' => '',);

// Contenu du tableau.
$contenuTableau = array(
	"champ 1", 1, 2,
	"champ 2", 3, 4,
	"champ 3", 5, 6,
	"champ 4", 7, 8,);
```

Commentaires

- **$proprietesTableau** Cette variable contient les propriétés du tableau, l'alignement, la marge, la couleur et l'épaisseur de bord;
- **$proprieteHeader** Cette variable contient les propriétés du header, couleurs, taille, font, alignement du texte ; couleur et taille des bordures;
- **$contenuHeader** Cette variable contient les largeurs des colonnes et leurs contenus;
- **$proprieteContenu** Du même type que $proprieteHeader, cette variable contient les propriétés du reste du tableau (elles peuvent différer...);
- **$contenuTableau** Cette variable contient le contenu du tableau, pour X colonnes et Y lignes, il faut X*Y items dans ce tableau.

3) Enfin, on génère le tableau et on affiche le résultat.

```
// D'abord le PDF, puis les propriétés globales du
tableau.
// Ensuite, le header du tableau (propriétés et
données) puis le contenu (propriétés et données)
$PDF->drawTableau($PDF, $proprietesTableau,
$proprieteHeader, $contenuHeader, $proprieteContenu,
$contenuTableau);

$PDF->Output();
```

4. Conclusion

Tout d'abord, merci à Yogui et à Julp pour les relectures !!!

La classe phpToPDF est libre d'utilisation mais tenez-moi au courant de vos mises à jour.

Retrouvez l'article de jc-cornic en ligne : Lien18

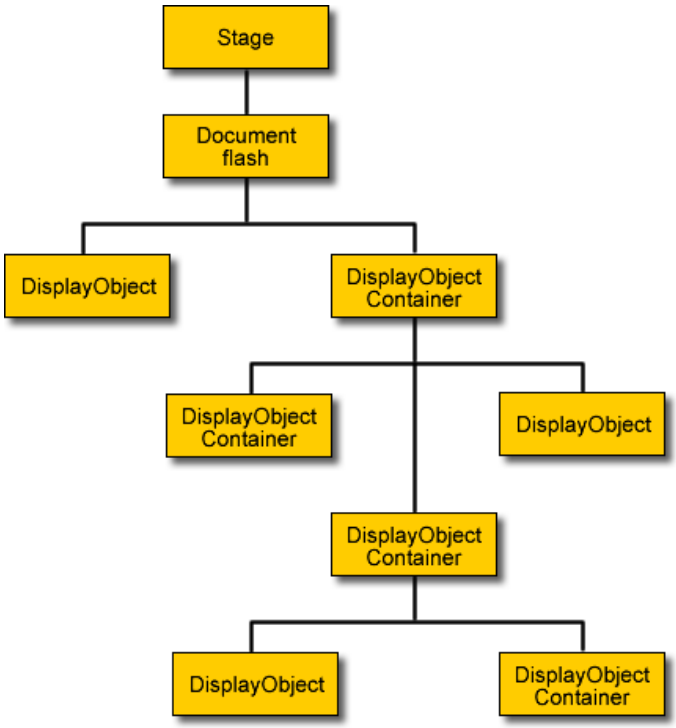
La nouvelle architecture du framework ActionScript 3

1. L'architecture

Dans la nouvelle architecture, une application flash possède une liste d'affichage ("display list"). Cette liste contient tous les éléments visibles. Ces éléments se repartissent en 2 types :

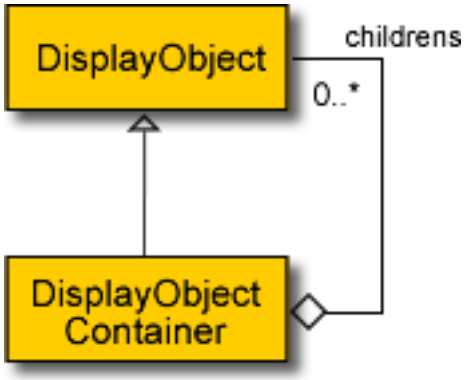
- **DisplayObject** : élément visible qui se caractérise par sa position, ses dimensions, etc.
- **DisplayObjectContainer** : zone rectangulaire qui peut

contenir aussi bien des DisplayObjectque des DisplayObjectConteneur.



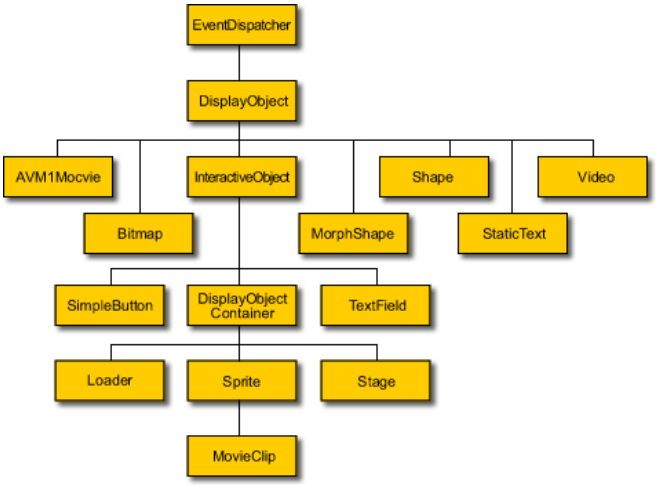
En regardent cette liste d'affichage, nous pourrions la comparer à un arbre dans lequel les feuilles sont les DisplayObjects et les noeuds sont les DisplayObjectContainers.

D'ailleurs cette structure est basée sur le design pattern Composite :



A la base de cette liste d'affichage, nous trouvons le type Stage qui est le conteneur principal. Nous pouvons le comparer au _root de cette application. L'accés a ce conteneur principal se fait à l'aide de la propriété stage de tous les DisplayObjects .

Mais où sont dans tout ça, les MovieClip, les TextField, etc. Voici un schéma qui nous les situe :



Nous remarquerons au passage que tout est EventDispatcher, ce qui signifie que tous les éléments de ce schèma peuvent émettre des évènements.

Retrouvez l'article complet d'Olivier Bugalotto accompagné d'un exemple en ligne : Lien19

DotNet

Les derniers tutoriels et articles

Créer et utiliser vos propres attributs pour paramètrer votre code

A travers cet article nous allons découvrir le système des attributs de dotnet, voir à quoi ils servent, et comment créer ses propres attributs

1. Les attributs ? Ké za ko ?

1.1. C'est quoi ?

Un attribut est un petit morceau de code permettant de paramétrer un assembly, une classe, un constructeur, un délégué, une énumération, un événement, un champ, une interface, une méthode, un paramètre, une propriété, une valeur de retour, une structure ou un autre attribut, etc ...
Ils ont une syntaxe praticulière et ne peuvent pas être confondu avec d'autres instructions du langage.

Une classe avec un attribut personnel

```
[MyAttribut(true)]
public class A{

}
```

1.2. A quoi ça sert ?

Les attributs servent donc à paramétrer une classe ou autre, afin que les appelant puissent facilement modifier leurs comportements en fonctions de ces paramètres.
Concrètement nous retrouvons les attributs dans beaucoup de classes de l'espace de nom System.Windows.Forms. Les attributs servent par exemple à définir l'image associée à un composant dans les boites à outils de nos EDI préférés.
Un autre attribut que nous rencontrons assez souvent est l'attribut Serializable. Dans ce cas l'attribut sert à "marquer" une classe comme étant sérialisable par le framework.

2. Utiliser les attributs

Maintenant que nous savons ce qu'est un attribut, voyons comment nous pouvons les exploiter dans nos programmes. L'utilisation des attributs est étroitement liée à la reflexion.
En effet c'est grâce à la réflexion que vous allez pouvoir déterminer si une classe, une propriété, un membre possède un ou plusieurs attributs.

Obtenir les attributs d'une classe

```
MyClass myClass = new MyClass();

Type cType = myClass.GetType();
object[] atts = cType.GetCustomAttributes(false); //On
obtient les attributs sans prendre en compte l'héritage

for(int i = 0; i  atts.Length; i++ ){
   if( atts[i] is CustomAttribut ){
              Console.WriteLine(
((CustomAttribut)atts[i]).MyValue );
   }
}
```

3. Comment créer ses propres attributs ?

Rien n'est plus simple que de créer ces propres attributs. ;-)
Pour cela il suffit de créer une classe dérivant de la classe Attribute de faire terminer le nom de sa classe par Attribute.

Création d'un attribut personnalisé

```
public class CustomAttribute : Attribute{
        private string myValue;

        public CustomAttribute(string myValue){
                this.myValue = myValue;
        }

        public string MyValue{
                get{
                        return this.myValue;
                }
        }
}
```

Cet attribut s'utilise ainsi :

Utilisation du CustomAttribut

```
[Custom("une valeur")]
public class UneClasse{

}
```

La valeur définie dans l'attribut lors de son utilisation est obligatoire à cause du constructeur défini la classe CustomAttribute.

3.1 Définition de la porté d'un attribut

Les attributs peuvent être conçu de façon a ce qu'il ne soit valable que sur un certain type de structure de code.
Ainsi un attribut peut être dédié spécialement aux propriétés, aux classes, aux méthodes, etc...
Afin de rendre un attribut spécialisé pour une structure de code il suffit de lui attribuer un ... attribut.

```
[AttributeUsage(AttributeTargets.Property)]
public class CustomAttribut{
        private string myValue;

        public CustomAttribute(string myValue){
                this.myValue = myValue;
        }

        public string MyValue{
                get{
                        return this.myValue;
                }
        }
}
```

L'énumération AttributeTargets peut prendre une multitude de valeur. Je vous invite à voir la msdn afin de connaître les possibilités.
Liste des enumérations de l'énumération AttributeTargets : Lien20

Comment faire pour définir un attribut pouvant être utilisé à la fois sur une propriété et une méthode par exemple ? Il suffit simplement de faire une addition binaire entre toutes les valeurs souhaitées.

```
[AttributeUsage(AttributeTargets.Property |
AttributeTargets.Method)]
```

3.2. L'héritage des attributs

Une chose à laquelle on ne pense pas forcément c'est le problème de l'héritage. En effet que ce passe-t-il quand nous créons une classe qui hérite d'une autre qui définie par exemple des propriétés paramétrées avec des attributs ?
Eh bien là aussi le framework nous permet de personnaliser le comportement de l'héritage des attributs.

```
/// <summary>
/// Classe définissant l'attribut personnalisé
/// </summary>
[AttributeUsage(AttributeTargets.Property,
Inherited=true)]
public class DisplayAttribute : Attribute{
    private bool display = false;

    /// <summary>
    /// Constructeur par défaut
    /// </summary>
    public DisplayAttribute(bool display){
        this.display = display;
    }

    /// <summary>
    /// Obtient ou défini une valeur indiquant si la
       propriété suivi a été modifié
    /// </summary>
    public bool Display{
        get{
            return this.display;
        }
    }
}
```

Le paramètre nommé **Inherited** permet de définir si l'attribut peut affecter les membres hérités.

Exemple :

```
public class Information{

        [Display(true)]
        public string Name{
                get{
                        return this.name;
                }

                set{
                        this.name = value;
                }
        }
}
```

```
public class InformationsSpecifiques : Information{

}
```

Dans cette exemple la classe InformationsSpecifiques hérite de la classe Information qui défini l'attribut **Display** à vrai pour la propriété **Name**. Que ce passe-t-il lorsqu'un objet de type InformationsSpécifique est examiné pour en extraire les attributs ?
Dans notre cas la propriété **Name** définie par la classe Information va être examineé et l'attribut **Display** va être détecté car nous avons défini le paramètre **Inherited** à **true**.
Au niveau du code de la détection y a t-il un moyen d'obtenir les attributs dont la propriété **Inherited** à **false** ?
Contrairement à toute attente oui ... En effet le fait de paramétrer un attribut avec **Inherited=false**, n'empêche pas les "inspecteurs" d'obtenir la valeur de l'attribut avec le code suivant :

```
pi.GetCustomAttributes(typeof(DisplayAttribute), true);
```

En effet le fait de positionner le second argument de **GetCustumAttributes** (inherited) à "vrai" permet d'obtenir les attributs hérités.

3.3. Instance multiple des attributs

Une autre question qui se pose est : "Est-il possible d'avoir plusieurs fois le même attribut sur le même élément ?".
Là encore la réponse nous est donnée par un paramètre nommé : AllowMultiple. Si ce paramètre est défini à true vous pourrez mettre deux fois l'attribut, sinon non.

```
public class Information{

        [Author("dev01")]
        [Author("author")]
        public string Name{
                get{
                        return this.name;
                }

                set{
                        this.name = value;
                }
        }
}
```

4. Cas concret d'utilisation des attributs personnalisés

Tout ceci est un peu trop théorique. C'est pourquoi nous allons voir un exemple d'utilisation des attributs personnalisés à travers un système permettant de paramétrer l'affichage des propriétés.

4.1. La classe à paramétrer

La classe ci-dessous sera la classe dont nous allons paramétrer l'affichage.

```
public class Informations{
   private string name;
   private string firstName;
   private string mail;

   public Informations(){

   }

   public string Name{
                get{
                        return this.name;
                }

                set{
                        this.name = value;
                }
   }
}
```

4.2. La classe d'attribut personnalisé

Cette classe défini notre attribut qui nous permettra de savoir si les informations de la classe doivent être visible ou non.

```
[AttributeUsage(AttributeTargets.Property,
Inherited=true, AllowMultiple = false)]
public class DisplayAttribute : Attribute{
        private bool display;

        public DisplayAttribute(bool display){
                this.display = display;
        }

        public bool Display{
                get{
                        return this.display;
                }

                set{
                        this.display = value;
                }
        }
}
```

L'attribut que nous avons défini prend un paramètre dans le constructeur afin de définir si la propriété doit être affichée ou non.
La propriété qui nous interesse ici est Display.

4.3. Utilisation de notre attribut personnalisé

Voyons maintenant comment nous allons nous servir de notre attribut.

```
public class Informations{
   private string name;
   private string firstName;
   private string mail;

   public Informations(){

   }

   [Display(true)]
   public string Name{
                get{
                        return this.name;
                }

                set{
                        this.
                        this.name = value;
                }
   }

   [Display(true)]
   public string FirstName{
                get{
                        return this.firstName;
```

```
            }

            set{
                this.firstName = value;
            }
    }

    public string Mail{
            get{
                return this.mail;
            }

            set{
                this.mail = value;
            }
    }

}
```

L'utilisation de notre attribut dans la classe à paramétrer est simple.
Il nous suffit de le définir à true pour dire que nous voulons que la propriété soit affichée.

4.4. La classe de gestion de l'affichage des propriétés

Cette classe permet d'afficher ou non une propriété en fonction de l'attribut Display.

```
private void UpdateInformations(){
            this.listViewInformations.Clear();
            this.listViewInformations.Columns.Clear();

            if( this.informations != null ){
                PropertyInfo[] pis =
this.informations.GetType().GetProperties();
                foreach (PropertyInfo pi in pis){
                    foreach (object o in
       pi.GetCustomAttributes(typeof(DisplayAttribute),
       false)){
                        if(
                       ((DisplayAttribute)o).Display){
                            this.listViewInformations.C
                              olumns.Add(pi.Name);
                        }
                    }
                }

                List<string> val = new List<string>();
                for (int i = 0; i <
this.listViewInformations.Columns.Count; i++){
                    val.Add(this.informations.GetType()
                      .GetProperty(
        this.listViewInformations.Columns[i].Text).
        GetGetMethod().Invoke(this.informations,
               null).ToString()
                      );
                }
                this.listViewInformations.Items.Add(new
               ListViewItem(val.ToArray()));
            }
        }
```

Comme vous le voyez l'utilisation des attributs s'appuie beaucoup sur la réflexion.
Dans le détail cette fonction obtient la totalité des propriétés de la classe, et extrait pour chaque propriété la liste des attributs de type **DisplayAttribute**.
Ensuite si la propriété doit être affiché alors le nom de la propriété est ajouter comme colonne dans le ListView.
La suite est assez simple, il s'agit de remplir la ligne avec les informations.

5. Conclusion

L'utilisation des attributs personnalisés est très utile afin de paramétrer une classe au moment du développement pour influencer le comportement de composant utilisant cette classe.

C'est un moyen simple et efficace de fournir des informations sur la façon d'utiliser une classe.

Retrouvez l'article de Vicent Lainé : Lien21

Créer et associer un type de fichier à son application

Le but de cet article est de pouvoir créer son propre type de fichier (avec une extension et une icône personnalisées) et de l'associer à son application en utilisant un projet de déploiement Visual Studio 2005.

1. Introduction

Vous venez de créer une application et celle-ci charge des fichiers de données personnalisés (avec une extension particulière). Ce que vous aimeriez maintenant c'est pouvoir associer ces fichiers à votre application et à une icône particulière. Ainsi, de même que lorsque vous faîtes un double clic sur un fichier PDF celui-ci s'ouvre avec Acrobat, un double clic sur un fichier dont vous aurez personnalisé le type l'ouvrira avec votre application.
Nous allons voir que cela peut se faire facilement en modifiant un peu le code de l'application et en paramétrant correctement la solution de déploiement au travers de Visual Studio 2005.

2. Création de l'application

2.1. Présentation de l'application

Nous allons illustrer ce tutoriel en créant une petite application. Celle-ci sera composée d'un textbox et d'un bouton. L'appui sur ce dernier enregistrera le contenu de la textbox dans un fichier texte avec une extension ".toto".

Si vous créez un fichier avec une extension ".toto", Windows lui associe une icône générique. Si vous faite un double clic dessus, Windows vous demande avec quel logiciel l'ouvrir. En effet, aucun logiciel ne lui a été associé. Notre but est qu'une icône soit associée à ce fichier et qu'un double clic dessus lance notre petite application et que celle-ci lise le fichier et affiche son contenu dans la textbox.

2.2. Le code

Voici tout d'abord le code associé à notre bouton :

```
private void buttonSave_Click(object sender, EventArgs
e){
    try {
        SaveFileDialog saveFileDialog1 = new
                SaveFileDialog();
```

```
        saveFileDialog1.Filter = "fichiers toto
        (*.toto)|*.toto|Tous les fichiers (*.*)|*.*";
        saveFileDialog1.FilterIndex = 1;
        saveFileDialog1.RestoreDirectory = true;

        if (saveFileDialog1.ShowDialog() ==
               DialogResult.OK){
            // Ecriture du fichier. On l'écrase s'il
               existe déjà
            using (System.IO.StreamWriter sw = new
               System.IO.StreamWriter(
               saveFileDialog1.FileName, false)){
                sw.WriteLine(textBox1.Text);
            }
        }
    }
    catch (Exception ex){
        MessageBox.Show("Erreur lors de la sauvegarde:
       " + ex.Message);
    }
}
```

Vraiment rien d'exceptionnel ici. On demande où l'utilisateur veut enregistrer le fichier grâce à un saveFileDialog (auquel on ajoute un filtre sur les fichiers d'extension ".toto"). On crée ensuite à cet endroit un nouveau fichier contenant le texte de la textbox.

Intéressons-nous maintenant au chargement des fichiers ".toto". Lorsque l'utilisateur fera un double clic (ou un clic droit, ouvrir) sur un de ces fichiers, le chemin de celui-ci sera passé argument de notre application. Il nous faut donc le récupérer pour pouvoir charger le fichier.

Il est possible d'obtenir la liste des arguments grâce à la méthode **GetCommandLineArgs()** de la classe Environment. Cette méthode revoit un tableau de **String** contenant la liste des arguments de l'application.

Attention !
Ce tableau d'argument n'est jamais vide. En effet, la première case du tableau contient le chemin de l'exécutable de notre application. Le chemin du fichier à charger sera donc dans la deuxième case (index 1).

Voici donc le code de chargement d'un fichier ".toto". Ce code est placé dans la méthode Load de la form de l'application, mais rien ne vous empêche de le mettre ailleurs.

```
private void MonAppli_Load(object sender, EventArgs e){
    try{
        //si le chemin du fichier est passé en argument
        if (Environment.GetCommandLineArgs().Length ==
               2){
            MessageBox.Show("Chargement du fichier: " +
               Environment.GetCommandLineArgs()[1]);
            string fileContents;
            //lecture du contenu du fichier
            using (System.IO.StreamReader sr = new
               System.IO.StreamReader(
               Environment.GetCommandLineArgs()[1])){
                fileContents = sr.ReadToEnd();
            }
            //on met le contenu du fichier dans la
               textbox
            textBox1.Text = fileContents;
        }
    }
    catch (Exception ex){
        MessageBox.Show("Erreur lors du chargement: " +
               ex.Message);
    }
}
```

On vérifie qu'un chemin de fichier est bien passé en paramètre. Si c'est le cas, on le récupère avec **Environment.GetCommandLineArgs()[1]**. On lit ensuite ce fichier avec un **StreamReader** et on affiche son contenu dans la textbox.

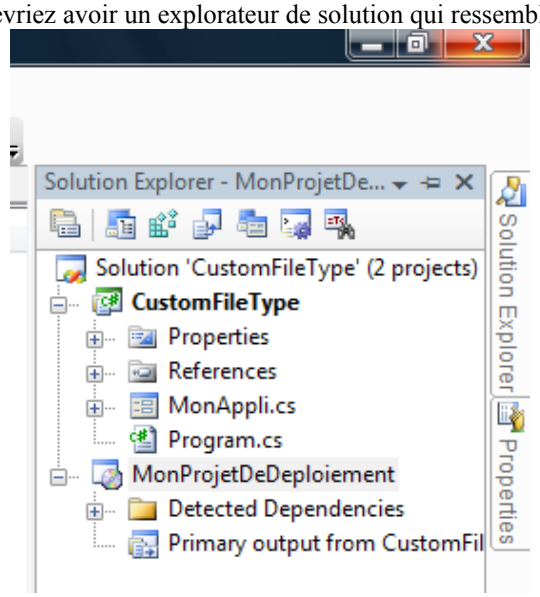
3. Le projet de déploiement

3.1. Ajouter un nouveau projet

Maintenant que notre application est capable d'ouvrir un fichier dont le chemin lui est passé en argument, il reste à s'occuper de l'association proprement dite.

Il nous faut tout d'abord ajouter un nouveau projet de type déploiement à notre solution. Pour cela, faites un clic droit sur le nom de la solution dans l'explorateur de solution et choisissez **nouveau projet**.

Configurez ensuite le projet de déploiement selon vos besoins, sans oublier d'y inclure notre petite application.
Vous devriez avoir un explorateur de solution qui ressemble à ça :



3.2. Configurer le projet de déploiement

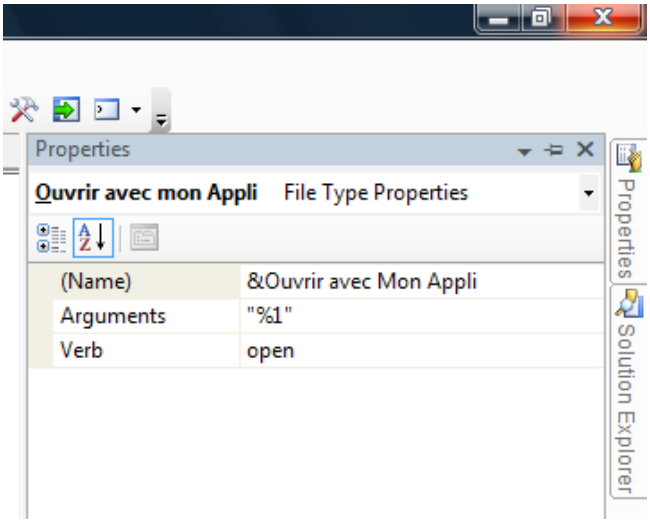
Une fois le nouveau projet ajouté, faîtes un clic droit sur son nom et choisissez **"Voir"**, **"Types de fichier"**. Cela va ouvrir un designer (en onglet) pour les types de fichier :

Dans ce nouveau designer, cliquez sur **"Types de fichier sur la machine cible"** et choisissez **"Ajouter un type de fichier"**.

Cela va ajouter un nouvel élément dans le designer. Sélectionnez-le et aller dans l'éditeur de propriétés pour les modifier.

Complétez maintenant les différents champs de ce nouveau type de fichier selon vos besoins. Remplissez le nom, la description, l'extension et assignez une icône à ce nouveau type de fichier. Dans le champ "commande", pointez vers l'application avec laquelle ouvrir ce type de fichiers.

Retournons maintenant au designer. Sous le nouveau type de fichier créé, cliquez sur le noeud "ouvrir".Aller ensuite dans la fenêtre de propriétés pour les éditer :



Ne touchez pas aux deux derniers champs, leur valeur par défaut nous suffit. Le champ **"Arguments"** spécifie que le chemin du fichier sera passé en argument de l'application. Le champ **"verbe"** spécifie l'action à effectuer avec le fichier. Le champ **"Nom"** indique le libellé affiché dans le menu contextuel lors d'un clic droit sur un fichier de ce type (voir image suivant). Modifiez-le à votre guise.

La configuration du projet de déploiement est maintenant terminée. Vous pouvez compiler le projet et le déployer pour tester votre nouveau type de fichier personnalisé.

4. Conclusion

Nous avons donc vu comment créer son propre type de fichier et l'associer à son application pour les ouvrir automatiquement.
Cela ne nécessite que peu de lignes de code et le paramétrage du projet de déploiement dans Visual Studio 2005 est vraiment très simple à réaliser.

Retrouvez l'article de Florian Casabianca avec plus de captures d'écran en ligne : Lien22

Les derniers tutoriels et articles

Ecriture de driver sous Linux grâce au Langage C

Dans cet article, vous verrez les bases pour créer vos propres drivers sous Linux, grâce au Langage C.

1. Avant-Propos

Tout d'abord sous Linux, il existe 2 modes différents, le mode noyau et le mode utilisateur.

- Espace noyau : tout est permis même le pire, on peut vite tout casser.
- Espace utilisateur : tout est protégé, les possibilités sont bridées.

La création de driver et de module utilise le mode noyau, alors faites attention !!!!

2. Mode noyau et module

2.1. Débogage en mode noyau

Le débogage en mode noyau peut se faire via plusieurs outils.

- En mode console, via **printk()** et **dmesg** pour voir les messages;
- Avec **kgdb**, nécessite une seconde machine reliée par câble série et possédant GDB-client pour déboguer la cible;
- Avec **kdb**, un débogeur noyau embarqué.

Il existe d'autres méthodes mais, ne les ayant jamais utilisées, je n'en parlerai pas.

Prototype :

```
int printk(const char *fmt, ...)
```

Exemple :

```
printk("<1> Hello World !\n");
```

Plusieurs niveaux de débogage sont définis dans **<linux/kernel.h>**

```
#define KERN_EMERG    "<0>"  /* système inutilisable */
#define KERN_ALERT    "<1>"  /* action à effectuer immédiatement*/
#define KERN_CRIT     "<2>"  /* conditions critiques */
#define KERN_ERR      "<3>"  /* conditions d'erreurs */
#define KERN_WARNING  "<4>"  /* message d'avertissement */
#define KERN_NOTICE   "<5>"  /* normal mais significatif */
#define KERN_INFO     "<6>"  /* informations */
#define KERN_DEBUG    "<7>"  /* messages de débugging */
```

Du coup cela devient :

```
printk(KERN_ALERT "Hello World !\n");
```

La commande **dmesg** permet d'afficher les messages de printk()

2.2. Chargement et déchargement de module

Un module possède un point d'entrée et un point de sortie.

- point d'entrée : **int xxx(void)**
- point de sortie : **void yyy(void)**

Où xxx et yyy sont ce que vous voulez. Pour dire au module que ces 2 fonctions sont le point d'entrée et le point de sortie, nous utilisons ces 2 macros.

- module_init(xxx);
- module_exit(yyy)

Dans la suite de l'article les points d'entrée et de sortie sont nommés **module_init()** et **module_exit()**.

Ces 2 fonctions sont automatiquement appelées, lors du chargement et du déchargement du module, avec **insmod** et **rmmod**.

Remarque : Un module ne possède pas de fonction **main()**.

La fonction **module_init()** doit s'occuper de préparer le terrain pour l'utilisation de notre module (allocation mémoire, initialisation matérielle...).

La fonction **module_exit()** doit quant à elle défaire ce qui a été fait par la fonction module_init().

Voici le source minimum d'un module

```
#include <linux/module.h>
#include <linux/init.h>

static int __init mon_module_init(void){
        printk(KERN_DEBUG "Hello World !\n");
        return 0;
}

static void __exit mon_module_cleanup(void){
    printk(KERN_DEBUG "Goodbye World!\n");
}

module_init(mon_module_init);
module_exit(mon_module_cleanup);
```

Vous pouvez ainsi appeler vos **init** et **cleanup** comme bon vous semble.

Pour compiler c'est très simple, il faut utiliser cette commande :
make

Avec le makefile suivant :

```
obj-m += module.o

default:
	make -C /lib/modules/$(shell uname -r)/build
	M=$(PWD) modules

clean:
	make -C /lib/modules/$(shell uname -r)/build
	M=$(PWD) clean
```

Chargement et déchargement :
$ insmod ./module.ko
$ lsmod
$ rmmod module.ko
$ dmesg

2.3. Description de module

Vous pouvez décrire vos modules à l'aide des différentes macros mises à votre disposition dans **<linux/module.h>**.

MODULE_AUTHOR(nom) : place le nom de l'auteur dans le fichier objet
MODULE_DESCRIPTION(desc) : place une description du module dans le fichier objet
MODULE_SUPPORTED_DEVICE(dev) : place une entrée indiquant le périphérique pris en charge par le module.
MODULE_LICENSE(type) : indique le type de licence du module

On peut obtenir ces informations avec la commande **modinfo nom_module**.

```
#define MODULE
#include <linux/module.h>
#include <linux/init.h>

MODULE_AUTHOR("skyrunner");
MODULE_DESCRIPTION("exemple de module");
MODULE_SUPPORTED_DEVICE("none");
MODULE_LICENSE("none");


static int __init mon_module_init(void){
    printk(KERN_DEBUG "Hello World !\n");
    return 0;
}

static void __exit mon_module_cleanup(void){
    printk(KERN_DEBUG "Goodbye World!\n");
}

module_init(mon_module_init);
module_exit(mon_module_cleanup);
```

2.4. Passage de paramètres

Comme vous vous en doutez, il serait bien utile de passer des paramètres à notre module.
une fonction et une macro sont donc disponibles pour cela

module_param(nom, type, permissions)
MODULE_PARM_DESC(nom, desc)

Plusieurs types de paramètres sont actuellement supportés pour le paramètre type : short (entier court, 2 octet), int (entier, 4 octets), long (entier long) et charp (chaînes de caractères).

Dans le cas de tableau, vous devez utiliser la fonction : module_param_array(name, type, addr, permission);
addr est l'adresse d'une variable qui contiendra le nombre d'élément initialisés par la fonction.

```
#include <linux/module.h>
#include <linux/init.h>

MODULE_AUTHOR("skyrunner");
MODULE_DESCRIPTION("exemple de module");
MODULE_SUPPORTED_DEVICE("none");
MODULE_LICENSE("none");

static int param;

module_param(param, int, 0);
MODULE_PARM_DESC(param, "Un paramètre de ce module");

static int __init mon_module_init(void){
    printk(KERN_DEBUG "Hello World !\n");
    printk(KERN_DEBUG "param=%d !\n", param);
    return 0;
}

static void __exit mon_module_cleanup(void){
    printk(KERN_DEBUG "Goodbye World!\n");
}

module_init(mon_module_init);
module_exit(mon_module_cleanup);
```

Test:
$insmod ./module.o param=2

Voilà, maintenant que vous connaissez les bases d'un module, nous allons voir la création d'un driver et les interactions possibles avec le mode utilisateur, et le matériel.

3. Driver en mode caractère

Un driver en mode caractère permet de dialoguer avec le périphérique, en échangeant des informations. Il existe d'autre driver, dit en mode bloc, qui échangent des données uniquement par bloc de données (disque dur par exemple).
Dans cet article, nous nous intéresserons uniquement aux drivers en mode caractère.

3.1. Ajout d'un driver au noyau

Lors de l'ajout d'un driver au noyau, le système lui affecte un nombre majeur. Ce nombre majeur a pour but d'identifier notre driver.

L'enregistrement du driver dans le noyau doit se faire lors de l'initialisation du driver (c'est à dire lors du chargement du module, donc dans la fonction **init_module()**), en appelant la fonction : **register_chrdev()**.

De même, on supprimera le driver du noyau (lors du déchargement du module dans la fonction **cleanup_module()**), en appelant la fonction : **unregister_chrdev()**.

Ces fonctions sont définies dans <linux/fs.h>

Prototypes :

```
int register_chrdev(unsigned char major, const char
```

```
*name, struct file_operations *fops);
int unregister_chrdev(unsigned int major, const char
*name);
```

Ces fonctions renvoient 0 ou >0 si tout se passe bien.

register_chrdev

- **major** : numéro majeur du driver, 0 indique que l'on souhaite une affectation dynamique.
- **name** : nom du périphérique qui apparaîtra dans /proc/devices
- **fops** : pointeur vers une structure qui contient des pointeurs de fonction. Ils définissent les fonctions appelées lors des appels systèmes (open, read...) du côté utilisateur.

unregister_chrdev

- **major** : numéro majeur du driver, le même qu'utilisé dans **register_chrdev**
- **name** : nom du périphérique utilisé dans **register_chrdev**

```
static struct file_operations fops = {
    read : my_read_function,
    write : my_write_function,
    open : my_open_function,
    release : my_release_function /* correspond à close
*/
};
```

Une autre façon de déclarer la structure **file_operations**, qui reste plus portable et "correcte" est la suivante

```
struct file_operations fops ={
    .read = my_read_function,
    .write = my_write_function,
    .open = my_open_function,
    .release = my_release_function  /* correspond a
close */
};
```

Certaines méthodes non implémentées sont remplacées par des méthodes par défaut. Les autres méthodes non implémentées retournent **-EINVAL.**

3.2. Implémentation des appels systèmes

```
static ssize_t my_read_function(struct file *file, char
*buf, size_t count, loff_t *ppos){
    printk(KERN_DEBUG "read()\n");
    return 0;
}

static ssize_t my_write_function(struct file *file,
const char *buf, size_t count, loff_t *ppos){
    printk(KERN_DEBUG "write()\n");
    return 0;
}

static int my_open_function(struct inode *inode, struct
file *file){
    printk(KERN_DEBUG "open()\n");
    return 0;
}

static int my_release_function(struct inode *inode,
struct file *file){
    printk(KERN_DEBUG "close()\n");
    return 0;
}
```

Ces fonctions renvoient 0 (ou >0) en cas de succès, une valeur négative sinon.

La structure file définie dans **<linux/fs.h>** représente un fichier ouvert et est créee par le noyau sur l'appel système **open()**. Elle est transmise à toutes fonctions qui agissent sur le fichier, jusqu'à l'appel de **close().**

Les champs les plus importants sont :

- **mode_t f_mode** : indique le mode d'ouverture du fichier
- **loff_t f_pos** : position actuelle de lecture ou d'écriture
- **unsigned int f_flags** : flags de fichiers (O_NONBLOCK...)
- **struct file_operations *f_op** : opérations associées au fichier
- **void *private_data** : le driver peut utiliser ce champ comme il le souhaite

Un fichier disque sera lui représenté par la structure inode. On n'utilisera généralement qu'un seul champ de cette structure :
kdev_t i_rdev : le numéro du périphérique actif

Ces macros nous permettrons d'extraire les nombres majeurs et mineurs d'un numéro de périphérique :
int minor = MINOR(inode->i_rdev);
int major = MAKOR(inode->i_rdev);

3.3. Méthodes open et release

En général, la méthode open réalise ces différentes opérations :

- Incrémentation du compteur d'utilisation
- Contrôle d'erreur au niveau matériel
- Initialisation du périphérique
- Identification du nombre mineur
- Allocation et remplissage de la structure privée qui sera placée dans file->private_data

Le rôle de la méthode release est tout simplement le contraire

- Libérer ce que open a alloué
- Éteindre la périphérique
- Décrémenter le compteur d'utilisation
-

Dans les noyaux actuels, le compteur d'utilisation est ajusté automatiquement, pas besoin de s'en occuper.

```
#include <linux/module.h>
#include <linux/init.h>
#include <linux/fs.h>

MODULE_AUTHOR("skyrunner");
MODULE_DESCRIPTION("premier driver");
MODULE_SUPPORTED_DEVICE("none");
MODULE_LICENSE("none");

static int major = 254;

module_param(major, int, 0);
MODULE_PARM_DESC(major, "major number");

static ssize_t my_read_function(struct file *file, char
*buf, size_t count, loff_t *ppos){
    printk(KERN_DEBUG "read()\n");
    return 0;
}

static ssize_t my_write_function(struct file *file,
```

```
const char *buf, size_t count, loff_t *ppos){
    printk(KERN_DEBUG "write()\n");
    return 0;
}

static int my_open_function(struct inode *inode, struct
file *file){
    printk(KERN_DEBUG "open()\n");
    return 0;
}

static int my_release_function(struct inode *inode,
struct file *file){
    printk(KERN_DEBUG "close()\n");
    return 0;
}

static struct file_operations fops = {
    read : my_read_function,
    write : my_write_function,
    open : my_open_function,
    release : my_release_function /* correspond a close
*/
};

static int __init mon_module_init(void){
    int ret;

    ret = register_chrdev(major, "mydriver", &fops);

    if(ret < 0){
        printk(KERN_WARNING "Probleme sur le major\n");
        return ret;
    }

    printk(KERN_DEBUG "mydriver chargé avec succès\n");
    return 0;
}

static void __exit mon_module_cleanup(void){
    int ret;

    ret = unregister_chrdev(major, "mydriver");

    if(ret < 0){
        printk(KERN_WARNING "Probleme unregister\n");
    }

    printk(KERN_DEBUG "mydriver déchargé avec
succès\n");
}

module_init(mon_module_init);
module_exit(mon_module_cleanup);
```

Une fois le driver compilé et chargé, il faut tester son lien, avec le mode utilisateur, et plus particulièrement les appels systèmes.

Tout d'abord, nous devons créer son fichier spécial : **mknod /dev/mydriver c 254 0**

Nous pouvons désormais effectuer notre premier test : **$ cat mydriver.c > /dev/mydriver**

Maintenant vous pouvez vérifier en tapant dmesg, et voir les appels à **open(), write()** et **release()**.

Vous pouvez aussi créer votre propre programme qui ouvre le périphérique et envoi une donnée, puis le referme.

Exemple:

```
#include <stdio.h>
#include <unistd.h>
#include <errno.h>

int main(void){
        int file = open("/dev/mydriver", O_RDWR);

        if(file < 0){
                perror("open");
                exit(errno);
        }

        write(file, "hello", 6);

        close(file);

        return 0;
}
```

3.4. Allocation mémoire

En mode noyau, l'allocation mémoire se fait via la fonction **kmalloc**, la désallocation, elle se fait via **kfree**.

Ces fonctions, sont très peu différentes des fonctions de la bibliothèque standard. La seule différence, est un argument supplémentaire pour la fonction kmalloc() : la priorité.

Cet argument peut-être :

- **GFP_KERNEL** : allocation normale de la mémoire du noyau
- **GFP_USER** : allocation mémoire pour le compte utilisateur (faible priorité)
- **GFP_ATOMIC** : alloue la mémoire à partir du gestionnaire d'interruptions

```
#include <linux/slab.h>

buffer = kmalloc(64, GFP_KERNEL);
if(buffer == NULL){
    printk(KERN_WARNING "problème kmalloc !\n");
    return -ENONEM;
}
kfree(buffer), buffer = NULL;
```

3.5. Méthodes read et write

Ces 2 méthodes renvoient le nombre d'octets, lus pour **read** et écrits pour **write**.

```
static int buf_size = 64;
static char *buffer;

static ssize_t my_read_function(struct file *file, char
*buf, size_t count, loff_t *ppos){
        int lus = 0;

        printk(KERN_DEBUG "read: demande lecture de %d
        octets\n", count);
        /* Check for overflow */
        if (count <= buf_size - (int)*ppos)
                lus = count;
        else  lus = buf_size - (int)*ppos;
        if(lus)
                copy_to_user(buf, (int *)p->buffer +
                (int)*ppos, lus);
        *ppos += lus;
        printk(KERN_DEBUG "read: %d octets reellement
```

```
lus\n", lus);
        printk(KERN_DEBUG "read: position=%d\n",
(int)*ppos);
        return lus;
}

static ssize_t my_write_function(struct file *file,
char *buf, size_t count, loff_t *ppos){
        int ecrits = 0;
        int i = 0;

        printk(KERN_DEBUG "write: demande ecriture de %d
        octets\n", count);
        /* Check for overflow */
        if (count <= buf_size - (int)*ppos)
                ecrits = count;
        else  ecrits = buf_size - (int)*ppos;

        if(ecrits)
                copy_from_user((int *)p->buffer +
                (int)*ppos, buf, ecrits);
        *ppos += ecrits;
        printk(KERN_DEBUG "write: %d octets reellement
        ecrits\n", ecrits);
        printk(KERN_DEBUG "write: position=%d\n",
        (int)*ppos);
        printk(KERN_DEBUG "write: contenu du buffer\n");
        for(i=0;i<buf_size;i++)
                printk(KERN_DEBUG " %d", p->buffer[i]);
        printk(KERN_DEBUG "\n");
        return ecrits;
}
```

L'allocation mémoire de **buffer** se fera lors du chargement du module.
buffer = kmalloc(buf_size, GFP_KERNEL);

Pour comprendre l'utilisation de **buffer** et de **ppos**, vous n'avez qu'à faire plusieurs essais en mode utilisateur, avec par exemple des appels à lseek.

Les fonctions **copy_from_user** et **copy_to_user** servent à transférer un buffer depuis/vers l'espace utilisateur.

```
copy_from_user(unsigned long dest, unsigned long src,
unsigned long len);
copy_to_user(unsigned long dest, unsigned long src,
unsigned long len);
```

3.6. Méthode ioctl

Dans le cadre d'un pilote, la méthode **ioctl** sert généralement à contrôler le périphérique.

Cette fonction permet de passer des commandes particulières au périphérique. Les commandes sont codées sur un entier et peuvent avoir un argument. L'argument peut-être un entier ou un pointeur vers une structure.

Prototype :

```
int ioctl(struct inode *inode, struct file *file,
unsigned int cmd, unsigned long arg);
```

Cette fonction est définie dans **<linux/fs.h>**.

La fonction **ioctl**, côté utilisateur a ce prototype:

```
int ioctl(int fd, int cmd, char *argp);
```

Une fonction type ressemble à :

```
int my_ioctl_function(struct inode *inode, struct file
*file, unsigned int cmd, unsigned long arg){
    int retval = 0;

    switch(cmd){
        case ... : ... break;
        case ... : ... break;
        default : retval = -EINVAL; break;
    }
    return retval;
}
```

Ne pas oublier de rajouter cette entrée dans la structure **file_operations**.

```
static struct file_operations fops = {
    read : my_read_function,
    write : my_write_function,
    open : my_open_function,
    release : my_release_function, /* correspond a
       close */
    ioctl : my_iotl_function;
};
```

Si la commande n'existe pas pour le pilote, l'appel système retournera **EINVAL**.

C'est au programmeur de choisir les numéros qui correspondent aux commandes.

Les numéros de commande, doivent être uniques dans le système, afin d'éviter l'envoi d'une commande au "mauvais" périphérique, il existe pour cela une méthode sûre pour choisir ses numéros.

Depuis la version 2.4, les numéros utilisent quatre champs de bits : **TYPE**, **NOMBRE**, **SENS** et **TAILLE**.

Le fichier header **<asm/ioctl.h>**, inclus dans **<linux/ioctl.h>**, définit des macros qui facilitent la configuration des numéros de commande.

_IO(type, nr) où type sera le nombre magique (voir le fichier ioctl-number.txt)
_IOR(type, nr, dataitem) // sens de transfert : Lecture
_IOW(type, nr, dataitem) // sens de transfert : Écriture
_IOWR(type, nr, dataitem) // sens de transfert : Lecture / Écriture

Une définition des numéros de commande ressemblera ainsi à ça :

```
...
#define SAMPLE_IOC_MAGIC 't'
#define GETFREQ _IOR(SAMPLE_IOC_MAGIC, 0, int)
#define SETFREQ _IOW(SAMPLE_IOC_MAGIC, 1, int)
```

Ceci peut être défini dans un fichier .h qui sera inclus, dans le code du driver, et dans le code du programme pilotant le driver.

```
#include "driver.h"
...

int my_ioctl_function(struct inode *inode, struct file
*file, unsigned int cmd, unsigned long arg){
    int retval = 0;

    switch(cmd){
        case SETFREQ : ... break;
        case GETFREQ : ... break;
        default : retval = -EINVAL; break;
    }
    return retval;
```

```
}
```

utilisateur.c

```
#include "driver.h"
...

ioctl(file, SETFREQ, 100);
```

3.7. Gestion d'interruptions

Une interruption est un signal envoyé par un matériel et qui est capable d'interrompre le processeur.

Notre pilote, met donc en place un gestionnaire d'interruptions pour les interruptions de son périphérique.

Un gestionnaire d'interruptions est déclaré par cette fonction : **request_irq().**

Il est libéré par cette fonction : **free_irq()**.

Les prototypes définis dans **<linux/sched.h>** sont les suivants:

```
int request_irq(unsigned int irq,
                void (*handler) (int, void , struct
pt_regs *),
                unsigned long flags /* SA_INTERRUPT ou
SA_SHIRQ */,
                const char *device, void *dev_id);

void free_irq(unsigned int irq, void *dev_id);
```

Quand l'interruption qui a pour numéro **irq** se déclenche, la fonction **handler()** est appelée.

Le champ **dev_id** sert à identifier les périphériques en cas de partage d'IRQ (SA_SHIRQ).

La liste des IRQ déjà déclarées est disponible dans **/proc/interrupts.**

Le gestionnaire d'interruptions peut-être déclaré à 2 endroits différents :

- au chargement du pilote (**module_init()**)
- à la première ouverture du pilote par un programme (**open()**), il faut alors utiliser le compteur d'utilisation et désinstaller le gestionnaire au dernier **close()**.

Le driver peut décider d'activer ou de désactiver le suivi de ses interruptions. Il dispose de 2 fonctions pour cela, définies dans **<linux/irq.h>**.

```
void enable_irq(int irq);
void disable_irq(int irq);
```

Cela peut-être utile pour ne pas être interrompu par une interruption, lors d'une interruption.

4. Conclusion

Vous venez donc de voir que la conception de driver sous Linux, n'est pas des plus difficiles, à vous maintenant d'apprendre plus tant au niveau informatique qu'électronique pour concevoir des drivers intéressants.

Retrouvez l'article de Benjamin Roux en ligne : Lien23

Vu dans la FAQ C++

Qu'est-ce que l'auto-affectation ?

Une auto-affectation a lieu quand quelqu'un affecte un objet à lui-même.

```
#include "Fred.hpp" // Déclaration de la classe Fred

void userCode(Fred& x){
    x = x; // Auto-affectation
}
```

Bien évidemment, personne n'écrit du code pareil, mais parce que des pointeurs ou des références distinctes peuvent désigner le même objet (c'est l'aliasing), des auto-affectations peuvent avoir lieu derrière votre dos.

```
void userCode(Fred& x, Fred& y){
    x = y; // C'est une auto-affectation si &x == &y
}

int main(){
    Fred z;
    userCode(z, z);

    return 0;
}
```

Pourquoi parle-t-on de l'auto-affectation ? Parce qu'elle peut être dangereuse. Imaginez une classe gérant un pointeur brut, et son opérateur d'affectation :

```
class MaClasse{
private :

    Ressource* ptr;

public :

    MaClasse& operator =(const MaClasse& Other){
        // Destruction de ptr
        delete this->ptr;

        // Réallocation et affectation de ptr
        this->ptr = new int(*Other.ptr);

        return *this;
    }
};
```

Dans le cas d'une auto-affectation, this et Other pointent vers la même instance, et donc vers le même ptr. Je vous laisse imaginer ce qu'il se passe lorsqu'on essaye de lire Other.ptr alors qu'il vient d'être détruit à la ligne précédente.

Pour éviter les problèmes d'auto-affectation, ou simplement pour tenter d'optimiser le code, on ajoute souvent un simple test permettant de vérifier que les deux instances sont différentes ; dans le cas contraire on peut quitter sans effectuer d'affectation.

```
MaClasse& MaClasse::operator =(const MaClasse& Other){
    if (this != &Other){
        // Destruction de ptr
        delete this->ptr;
```

```
        // Réallocation et affectation de ptr
        this->ptr = new int(*Other.ptr);
    }

    return *this;
}
```

Mais attention ce code aussi est un piège : en effet nous ajoutons un test que l'on croit utile, mais qui sera dans 99.9% des cas effectué pour rien (n'oubliez pas que l'auto-affectation est tout de même très rare). D'autant plus que si vous écrivez correctement votre opérateur d'affectation, comme indiqué dans la question Comment écrire un opérateur d'affectation correct ?, les éventuels problèmes d'auto-affectation sont résolus automatiquement de manière élégante.

Quand utiliser / ne pas utiliser using namespace ?

Utiliser **using namespace xxx**; indique au compilateur qu'il a le droit, quand il voit un nom dans le reste de la portée courante, de le rechercher dans l'espace de nom xxx, ce qui peut alléger le code, en permettant d'écrire :

```
using namespace std;

cout << hex << 42 << endl;
```

Au lieu de

```
std::cout << std::hex << 42 << std::endl;
```

Par contre, cette écriture est à proscrire dans des fichiers d'en-tête, du moins à portée de fichier. En effet, le but des espaces de nom est de permettre d'éviter des collisions de nom entre deux objets qui auraient le même nom, mais provenant de deux sources différentes (et donc classés dans deux espaces de nom différents). L'utilisation de **using** est un raccourci, mais il n'est possible que si on sait qu'il n'y a pas de conflits. Si ce n'est pas le cas, il faut obligatoirement utiliser le nom qualifié des objets.

Or, dans le cadre d'un fichier d'en-tête, on ne peut pas savoir dans quels contextes ce fichier sera utilisé. Et comme il n'existe pas de commande qu'on puisse insérer pour dire d'arrêter d'utiliser un **using**, on risque si on utilise cette écriture dans un fichier d'en-tête de provoquer un conflit chez un de ses clients, qui n'aura aucun recours pour le corriger.

Comment utiliser les flux pour afficher ou saisir mes objets ?

Pour injecter un objet de type quelconque dans un flux de sortie, il suffit de définir un opérateur << prenant en paramètre le flux, ainsi que l'objet à injecter.

```
#include <ostream>
#include <string>

class MaClasse{
    friend std::ostream& operator <<(std::ostream&,
const MaClasse&);
private :

    int Integer;
    std::string String;
};

std::ostream& operator <<(std::ostream& Stream, const
MaClasse& Obj){
    Stream << Obj.Integer << " " << Obj.String;
    return Stream; // N'oubliez pas de renvoyer le
flux, afin de pouvoir chaîner les appels
}

MaClasse Obj;
std::cout << "Ma classe : " << Obj << std::endl;
```

Notez qu'ici notre surcharge d'opérateur est déclarée amie de la classe, car elle a besoin d'accéder aux données privées de celle-ci. Cependant ce n'est pas obligatoire, notamment si les accesseurs adéquats ont été prévus dans la classe.

Le fonctionnement est le même pour l'extraction de valeurs à partir d'un flux d'entrée, via l'opérateur >> :

```
#include <istream>
#include <string>

class MaClasse{
    friend std::istream& operator >>(std::istream&,
       MaClasse&);

private :

    int Integer;
    std::string String;
};

std::istream& operator >>(std::istream& Stream,
MaClasse& Obj){
    Stream >> Obj.Integer >> Obj.String;
    return Stream; // N'oubliez pas de renvoyer le
flux, afin de pouvoir chaîner les appels
}

MaClasse Obj;
std::cin >> Obj;
```

Le fait de prendre en paramètre un ostream ou un istream permettra d'utiliser nos surcharges avec n'importe quel type de flux (stringstream, fstream, ...) puisque ceux-ci dérivent tous des classes ostream (pour les flux d'entrée) et istream (pour les flux de sortie).

Notez bien que ce genre de surcharge ne peut pas être membre de la classe, car celà impliquerait que l'opérande gauche soit l'objet et non le flux.

Enfin, si vous devez gérer l'injection et l'extraction sur une hiérarchie d'objets polymorphes, il faudra mettre en place une petite astuce (voir Comment surcharger correctement l'opérateur << pour afficher des objets polymorphiques ?).

Retrouvez l'entier de la FAQ C++ : Lien32

Structure des fichiers OpenXML

Dans cet article, je vous propose de faire connaissance avec l'organisation interne des fichiers OpenXML, le nouveau format de document Office de Microsoft.

1. Avant propos

OpenXML est le nouveau format de fichier adopté par les documents de la suite Office, à partir de la version 2007. Ce format, fruit de la collaboration de Microsoft, d'Intel et d'Apple, entre autres, est totalement libre de royalties, et sa pérennité et son indépendance vis-à-vis de tout éditeur sont garanties par son élévation au rang de norme par l'ECMA (le standard ISO devrait bientôt suivre).

OpenXML rompt radicalement avec les formats binaires propriétaires des précédentes versions d'Office (jusqu'à la version 2003, qui a amorcé le tournant), en adoptant XML, langage ouvert, comme format de stockage. Il sera désormais possible aux développeurs de lire, créer, modifier, afficher sur différents médias des documents Office, sans dépendre d'applications Microsoft, en utilisant des outils comme XSLT, SAX ou DOM, directement ou par l'intermédiaire de librairies OpenXML qui ne devraient pas tarder à apparaître.

OpenXML est une norme très riche, dont le corpus s'élève à plusieurs milliers de pages. Nous nous limiterons dans cet article d'initiation à l'étude de la section traitant de l'organisation du contenu des fichiers OpenXML intitulée Open Packaging Conventions (OPC), et plus précisément de l'application de ces conventions aux documents Office.

2. Anatomie d'un fichier OpenXML

2.1. Structure interne d'un fichier OpenXML

Les documents Office OpenXML sont des fichiers compressés selon le format Zip. On peut donc visualiser leur contenu en les décompressant à l'aide de n'importe quel utilitaire reconnaissant le format Zip.

Une fois décompressé, un fichier OpenXML révèle une kyrielle d'autres fichiers ; l'ensemble de ces fichiers est appelé, dans la terminologie OpenXML, un paquet (package). Ce paquet regroupe une collection de fichiers appelés parties (parts) répartis dans une arborescence dont ils constituent les feuilles. Cette structure éclatée tranche avec le format monolithique employé par Office 2003, dans lequel l'intégralité du document se retrouvait au sein du seul et même fichier XML. Ici, nous avons affaire à une structure modulaire, chaque partie ou répertoire contenant des parties étant un élément du fichier ZIP.

2.2. Les parties

OpenXML attribue à chaque partie un nom unique, coomposé du chemin logique menant de la racine du paquet au fichier constituant la partie proprement dite. Ainsi, de notre exemple de document nous pouvons tirer cette liste (partielle) de noms de parties :

- /[Content_Types].xml
- /_rels/.rels
- /word/document.xml
- /word/styles.xml
- /word/_rels/document.xml.rels
- /docProps/core.xml
- /word/media/image1.png
- ...

Les parties qui composent un fichier document OpenXML peuvent être réparties entre deux catégories :

1. Les parties qui contiennent les données (texte, images, sons, vidéos, etc.) qui constituent le document lui-même. Ces parties peuvent contenir des données définis dans OpenXML (du WordprocessingML dans notre exemple), d'autres données XML dont le schéma ne fait pas partie des spécifications OpenXML, des données binaires (objets OLE, images JPEG ou PNG, des vidéos AVI, etc.), du texte simple...
2. Les parties qui contiennent des informations concernant la structure interne du paquet, notamment le type de contenu des autres parties, et les liens logiques qu'elles ont entre elles ; ces parties contiennent des données XML dont le schéma est défini par la norme OpenXML (par les Open Packaging Conventions plus précisément)

OpenXML ne définit pas quels doivent être les noms des parties de la première catégorie, celles qui contiennent les données du document. Chaque application générant des fichiers OpenXML pouvant définir ses propres noms pour les parties, comment une application censée les lire peut-elle accéder à des parties dont elle ignore a priori le nom et le contenu ?

Cela est rendu possible grâce aux parties de la deuxième catégorie, qui contiennent les informations qui vont permettre de connaître précisément le rôle de chacune des autres parties du paquet, ainsi que les liens qui les unissent entre elles. Ces parties "spéciales" sont de deux types, le fichier des types de contenu et les fichiers de relations (relationships parts). Ces deux types de fichiers étant la clé de l'accès aux données du document OpenXML, nous allons les étudier en détail.

2.3. Le fichier des types de contenu

Le fichier des types de contenu d'un document Office OpenXML est la partie nommée **/[Content_Types].xml**.
Elle est placée à la racine du paquet et son nom est invariablement le même d'une instance de document OpenXML à l'autre, car défini par la norme.

Ce fichier contient un document XML qui recense toutes les

parties qui constituent le paquet, et associe à chacune un type MIME. Voici un extrait du contenu de ce fichier tiré de notre exemple de document :

Contenu de [Content_Types].xml

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Types
xmlns="http://schemas.openxmlformats.org/package/2006/content-types">
        <Override PartName="/word/footnotes.xml"
        ContentType="application/vnd.openxmlformats-
        officedocument.wordprocessingml.footnotes+xml"/>
        <Default Extension="png"
        ContentType="image/png"/>
        <Default Extension="emz" ContentType="image/x-
        emz"/>
        <Default Extension="xls"
        ContentType="application/vnd.ms-excel"/>
        <Override PartName="/word/comments.xml"
        ContentType="application/vnd.openxmlformats-
        officedocument.wordprocessingml.comments+xml"/>
        <Default Extension="rels"
        ContentType="application/vnd.openxmlformats-
        package.relationships+xml"/>
        <Default Extension="xml"
        ContentType="application/xml"/>
        <Override PartName="/word/document.xml"
        ContentType="application/vnd.openxmlformats-
officedocument.wordprocessingml.document.main+xml"/>
        <Override PartName="/word/numbering.xml"
        ContentType="application/vnd.openxmlformats-
        officedocument.wordprocessingml.numbering+xml"/>
        <Override PartName="/word/styles.xml"
        ContentType="application/vnd.openxmlformats-
        officedocument.wordprocessingml.styles+xml"/>
        <Override PartName="/word/endnotes.xml"
        ContentType="application/vnd.openxmlformats-
        officedocument.wordprocessingml.endnotes+xml"/>
        ...
</Types>
```

Le schéma de ce document XML est composé de l'élément principal **Types** et de ses deux éléments fils, **Default** et **Override**.

Default définit un type MIME par défaut, désigné par la valeur de l'attribut **ContentType**, pour les parties dont le nom se termine par l'extension contenue dans l'attribut Extension. Ainsi, dans notre exemple, toutes les parties dont le nom se termine par ".png" seront du type MIME "image/png", et celles se terminant par ".xml" seront du type MIME "application/xml".

L'élément **Override** signale qu'une partie a un type MIME autre que celui défini par défaut pour les parties ayant la même extension. Le nom de la partie bénéficiant de ce type MIME spécifique est contenu dans l'attribut **PartName**. Ainsi, par exemple, la partie nommée **/word/document.xml** est du type MIME application/vnd.openxmlformats-officedocument. wordprocessingml.document.main+xml, et non application/xml. Ce type MIME, ainsi que tous ceux des parties contenant du WordprocessingML, ont été définies spécifiquement par Microsoft pour les documents OpenXML issus d'Office.

Grâce à ce fichier de types de contenu, une application cliente peut déterminer de façon précise le contenu de chaque partie, sans avoir à faire de déduction hasardeuse à partir de son extension ou en recherchant dans son contenu une valeur "magique" indicatrice de la nature du fichier.

2.4. Les fichiers de relations

Ce sont des documents XML contenant un ensemble de relations, une relation étant un mappage entre une partie, la partie source, toujours implicite, et une autre partie du paquet (la partie cible). Les fichiers de relation constituent l'ossature du paquet.

L'emplacement de ces fichiers dans le paquet ainsi que leur nommage suivent plusieurs règles définies dans la spécification OpenXML. La première précise que le fichier de relation associé à une partie source doit se trouver dans un répertoire fils de celui qui contient la partie en question, et que ce répertoire doit impérativement se nommer _rels. La deuxième règle stipule que la fin du nom du fichier de relation doit être le même que celui de la partie associée, additionné de l'extension .rels.

Par exemple, le fichier de relation associé à la partie /word/document.xml sera localisé dans le paquet à l'emplacement /word/_rels, et se nommera /word/_rels/document.xml.rels.

2.4.1. Contenu d'un fichier de relation

Examinons le contenu d'un fichier de relation, par exemple /word/_rels/document.xml.rels :

Contenu de /word/_rels/document.xml.rels

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Relationships
xmlns="http://schemas.openxmlformats.org/package/2006/relationships">
        <Relationship Id="rId3"
Type="http://schemas.openxmlformats.org/officeDocument/
2006/relationships/settings" Target="settings.xml"/>
        <Relationship Id="rId7"
Type="http://schemas.openxmlformats.org/officeDocument/
2006/relationships/image" Target="media/image1.png"/>
        <Relationship Id="rId2"
Type="http://schemas.openxmlformats.org/officeDocument/
2006/relationships/styles" Target="styles.xml"/>
        <Relationship Id="rId5"
Type="http://schemas.openxmlformats.org/officeDocument/
2006/relationships/footnotes" Target="footnotes.xml"/>
        <Relationship Id="rId15"
Type="http://schemas.openxmlformats.org/officeDocument/
2006/relationships/footer" Target="footer1.xml"/>
        ...
</Relationships>
```

Le schéma de ce document comprend l'élément principal **Relationships** et ses éléments fils **Relationship**.

Chaque élément **Relationship** définit une relation entretenue entre la partie source (ici document.xml) et une autre partie du paquet. Les caractéristiques de la relation sont réparties entre les différents attributs de cet élément :

- L'attribut **Type** indique la nature de la relation. Elle se présente sous la forme d'une URI, dont la sémantique est propre à l'application à l'origine du document. Chaque éditeur implémentant OpenXML définit son propre jeu d'URI, en l'occurrence celles employées par les documents Office OpenXML sont toutes dotées du préfixe "http://schemas.openxmlformats.org/officeDocument/2006/relationships/"
- L'attribut **Target** indique la partie (ou la ressource) ciblée par la relation, sous la forme d'une URI toujours relative à la partie source
- L'attribut **Id** est l'identifiant unique de la relation ; il sera utilisé à l'intérieur de document.xml pour faire référence à cette relation, et au travers d'elle à la partie qu'elle cible
- L'attribut optionnel **TargetMode** indique si la cible de la relation est une partie située dans le paquet, ou bien, s'il est égal à "external", si la cible est une ressource externe au paquet ; en l'absence de cet attribut, la cible de la relation est interne au paquet

Pour formuler en langage naturel un exemple d'une de ces relations, la partie **/word/document.xml** a une relation nommée rId2 avec la partie **/word/styles.xml**, une relation dont le type est l'URI "http://schemas.openxmlformats.org/officeDocument/2006/relationships/styles". La lecture de la spécification OpenXML nous informe que cette URI indique que la ressource ciblée contient la définition des styles de caractères, de paragraphes et autres, de la partie source. Nous aurions pu nous en douter au vu du nom de la partie ciblée, mais l'URI nous le confirme de manière formelle et non ambiguë.

2.4.2. Le fichier de relation principal

Parmi les fichiers de relation que peut contenir un paquet, il en est un peu spécial puisqu'il n'est pas associé à une partie source, mais à la racine du paquet lui-même. Ce fichier de relation a la même structure que celle vue précédemment, et se nomme invariablement /_rels/.rels (notez que le nom de ce fichier suit les mêmes conventions que les autres fichiers de relation).

Examinons le contenu de ce fichier pour notre exemple de document :

Contenu de /_rels/.rels

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Relationships
xmlns="http://schemas.openxmlformats.org/package/2006/relationships">
	<Relationship Id="rId3"
Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/extended-properties"
Target="docProps/app.xml"/>
	<Relationship Id="rId2"
Type="http://schemas.openxmlformats.org/package/2006/relationships/metadata/core-properties"
Target="docProps/core.xml"/>
	<Relationship Id="rId1"
Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/officeDocument"
Target="word/document.xml"/>
</Relationships>
```

Parmi toutes les relations que peut contenir le fichier de relation principal, nous ne nous intéresserons dans ce tutoriel qu'à trois relations parmi les plus importantes :

- http://schemas.openxmlformats.org/officeDocument/2006/relationships/officeDocument Obligatoire. La cible de cette relation est la partie principale du document, celle qui contient le corps du document (word/document dans notre exemple)
- http://schemas.openxmlformats.org/package/2006/relationships/metadata/core-properties Pas obligatoire. La partie ciblée par cette relation contient les métadonnées communes à tous les documents Office, telles que la date de création, le titre, la description, le créateur, etc. (Dublin Core)
- http://schemas.openxmlformats.org/officeDocument/2006/relationships/extended-properties Par obligatoire. La partie ciblée par cette relation contient des propriétés spécifiques au type de document Office représenté par la partie principale du document ; s'il s'agit de WordprocessingML, les propriétés seront le nombre de pages, de caractères, de mots, de paragraphes, etc. S'il s'agit de SpreadsheetML, le nombre de feuilles de calculs, etc.

Le fichier de relation principal est donc le point d'entrée privilégié pour accéder à n'importe quelle partie du paquet, puisqu'à partir de la partie contenant le corps du document il est possible d'accéder aux parties annexes grâce à son fichier de relations. Si une partie annexe dispose elle-même d'un fichier de relations, on peut accéder encore à d'autres relations, et ainsi de suite, jusqu'à parcourir l'intégralité de l'arborescence du paquet.

2.5. Structure minimale d'un fichier OpenXML

Nous avons vu que les parties contenant les propriétés étendues et les métadonnées du document sont optionnelles, qu'en est-il des autres parties ? En fait, il en est de même de la majorité des autres parties qui figurent dans l'arborescence de notre document exemple ; ainsi, un document ne comprenant pas d'entête ni de pied de page n'aura pas de parties nommées /word/footer.xml et /word/header.xml. Quelles sont alors les parties qu'un fichier OpenXML Office devra comprendre a minima pour être lisible par les applications Office ? Elles sont au nombre de 3 :

- Le fichiers des types de contenu **/[Content_Types].xml**
- Le fichier de relation principal **/_rels/.rels**
- La partie contenant le corps du document (**/word/document.xml** pour un document Word)

3. Scénario de lecture d'un fichier Office OpenXML

Maintenant que nous avons examiné comment était structuré un fichier OpenXML, nous pouvons à présent établir l'enchaînement des tâches que devra effectuer une application pour accéder au contenu du document.

Ce processus se déroule en 4 phases :

1. Ouvrir le fichier de relation principal (_rels/.rels) et extraire la cible de la relation "http://schemas.openxmlformats.org/officeDocument/2006/relationships/officeDocument"
2. Ouvrir la partie [Content_Types].xml et se servir de la valeur obtenue précédemment comme clé pour rechercher le type MIME de la partie principale du document
3. (Optionnel) Lire les propriétés du document, les générales et les spécifiques, en ouvrant les parties correspondantes ciblées dans le fichier de relation principal ; les propriétés spécifiques seront interprétées conformément au type de document détecté à la phase 2
4. Lire la partie principale du document (le corps du document) identifié à la phase 1, et accéder aux autres parties constituantes du document grâce au fichier de relation de cette partie

Vous pourriez vous étonner que les deux premières phases soient nécessaires pour connaître le type de document Office auquel l'on a affaire, ne pourrait-on se baser pour cela sur l'extension du fichier, par exemple .docx pour un document Word ?
C'est effectivement possible, si vous maîtrisez totalement la production du document (c'est vous - votre application - qui le générez par exemple) et avez la garantie que l'extension correspond bien à son type. Si ce n'est pas le cas, si ce document vous provient par exemple d'un tiers sur lequel vous n'avez aucun contrôle, la consultation du fichier de relation principal et l'identification du type MIME constituent la méthode la plus robuste et la plus fiable pour connaître le type de document et le manipuler en conséquence.

D'autre part, vous pourriez avoir la tentation de vous passer des fichiers de relations pour atteindre les différentes parties du document, en partant du postulat que leur sont attribuées toujours les mêmes URI. Ainsi, **word/document.xml** contiendrait toujours le corps du document, **word/styles.xml** les styles employés dans

le document, etc. Encore une fois, cela ne se vérifiera que si vous maîtrisez totalement la génération de ces documents. Si ces documents sont générés par un processus hors de votre atteinte, ou même directement à partir d'Office, vous ne pouvez être absolument certain que cette nomenclature sera toujours la même. Plutôt que de faire des suppositions hasardeuses sur le rôle de chacune des parties du paquet, il est infiniment plus sage de se reposer sur les fichiers de relation pour reconstituer le puzzle et identifier les parties.

4. Conclusion

Les choix technologiques retenus par Microsoft pour son format OpenXML, XML et Zip, ouvrent la voie à pratiquement toutes les plateformes actuelles de développement pour sa manipulation. A présent, il ne vous reste plus qu'à mettre à profit toutes ces informations pour manipuler les fichiers OpenXML avec votre langage favori !

Retrouvez l'article d'Eric Grimois en ligne : Lien25

Qu'est-ce que XTech ?

XTech, la plus grande conférence annuelle des technologies XML.Une courte présentation

1. XTech, la conférence Européenne des technologies XML

Depuis 2005, en Europe,chaque année Xtech cherche à réunir les nombreux acteurs des technologies XML.

1.1. Sa genèse

Si XTech apparaît en 2005 à Amsterdam sous sa forme actuelle, elle ne surgit pas de nulle part, ses ancêtres sont la SGML Europe puis la XML Europe , soit plus de 25 ans d'histoire informatique.
Pourquoi ce changement ? Parce qu'en 2004 il est devenu évident ,par la popularisation de ces technologies et de l'extension de leur champ d'application, notamment WEB, qu'ils étaient difficiles de les contenir au "**peuple**" XML.

XTech est apparu en rajoutant 2 pistes à l'existant:

- Les développements des navigateurs
- Les formats ouverts

En effet, pour la première, l'émergence de Firefox a remis au centre des débats des sujets comme XUL,XAML,XHTML ou AJAX. On peut noté que XTech 2005 fut d'ailleurs un des principaux événements de la fondation Mozilla

Le web a encore permis le développement des formats , standards, ouverts régulièrement basé sur XML.

1.2. Ses buts

XTech est avant tout un forum, un espace d'échange , qui doit permettre à tous les acteurs de se mettre au courant des avancées, des nouveaux concepts mais aussi de partagée l'expérience accumulée dans ces domaines.

1.3. Son public

XTech cible deux publics essentiels:

- En premier lieu les développeurs, managers .... les utilisateurs de ces technologies.XTech leur permet d'échanger leur expérience, les dernières nouveautés mais aussi d'interroger les "faiseurs" de technologies.
- En second, ces mêmes "faiseurs" que ce soit de standard Web ou de plateforme logicielle.XTech leur permet aussi bien de se présenter, d'échanger entre eux, de se découvrir que d'avoir un retour d'expérience direct de leurs utilisateurs

Comme conférence, nous suivons un procédé pour le choix de papier qui inclut l'examen par les pairs des propositions. Chaque proposition est passée en revue anonyme par quatre ou cinq experts en matière de Web et évaluée. Nous choisissons alors le programme en conséquence, et chaque haut-parleur doit soumettre un papier. Ainsi nous ne sommes pas simplement une « exposition et ne disons pas » pour la dernière technologie, mais un forum sérieux pour des personnes travaillant avec le Web et les technologies de XML pour discuter et éditer leur travail.

1.5. Son organisation

XTech s'articulent autour de deux choses:

- des journées tutorial , destinés à l'apprentissage d'outils de technologies.....
- des présentations, pouvant mêlé plusieurs participants, évoquant nouvelles technologies comme retour d'expérience.

Les choix des présentations comme des tutoriaux suivent une sélection incluant l'examen par les pairs des propositions. Chaque proposition est passée en revue anonyme par quatre ou cinq experts en matière de Web. le programme est alors choisit en conséquence.Ce n'est pas une simple "exposition" des dernières technologies

2. Les sessions XTech

2.1. XTech 2005

XTech 2005 (Lien26) était intitulé "XML, the Web and Beyond". Elle prenait la place d' XML Europe et s'essayait à une vision plus large des technologies Web.
Pour la première les navigateurs, l'open data et des outils comme Apache et MySQL s'invitaient, et Creative Commons et la fondation Mozilla participaient aux conférences.

2.2 XTech 2006

XTech 2006 (Lien27) était intitulé "Building Web 2.0" et se focalisait sur les nouvelles technologies Web. Il y eut un "special Ajax Day" présentant les travaux les plus pointus dans le développement d'Ajax et une journée tutorial consacré à Ruby on Rails Nos présentations comprenaient Paul Graham, et des présentations par Yahoo! et Amazon de leurs plateforme.

2.3. XTech 2007

Cette année, à Paris, le thème d' XTech 2007 (Lien28) est « le Web omniprésent », voir comment le Web éclate l'environnement serveur-PC traditionnel, dans de diverses parties de nos vies, et par de divers dispositifs.

Retrouvez l'article d'Erwan Amoureux en ligne : Lien29

Optimisez vos requêtes DB2

Si vos requêtes ne sont pas correctement optimisées, cela peut conduire à des temps de réponses catastrophiques une fois que le serveur entre en pleine charge. DB2 dispose d'un optimiseur qui fait une grande partie du travail. Toutefois, il ne pourra être efficace que si nous respectons certaines règles.

1. Que fait l'optimizer de DB2

Pour optimiser nos requêtes, DB2 dispose d'un optimizer qui va grandement nous faciliter la tâche. Il va choisir automatiquement l'ordre des opérations internes pour exécuter au mieux la commande. Il choisira d'utiliser ou non un index et si oui, il choisira le plus approprié. Mais l'optimizer va plus loin car il est capable de réécrire votre requête pour la rendre plus performante. Par exemple, il pourrait transformer une sous-requête en jointure.

Malgré cela, il est préférable de coder directement au mieux votre commande SQL et de respecter dans la mesure du possible un certain nombre de règles que nous allons voir.

Attention, l'optimizer est différent selon les versions et les plateformes utilisées. C'est pourquoi même si les tests effectués dans cet article ne confirment pas nécessairement le gain avec les règles énoncées, il est malgré tout préférable de les respecter.

2. Les index

L'index permet la recherche rapide d'une valeur et le parcours ordonné selon l'ordre de l'index. Il permet également la sélection et, ou, le tri rapide des enregistrements si la sélection et ou le tri portent sur l'ordre de l'index.

Prenons comme exemple une table contenant 4 colonnes, C1, C2, C3, C4. Il existe un index portant sur les colonnes C1, C2.

```
SELECT * FROM table WHERE c1='xxx'
SELECT * FROM table WHERE c1>'xxx'
SELECT * FROM table WHERE c1= 'xxx' AND c2 = 'yyy'
```

Ces 3 commandes vont permettre un parcours via l'index.

Par contre,

```
SELECT * FROM table WHERE c2='yyy'
```

ne permet pas le parcours ordonné.

Toutefois, l'index permettra malgré tout d'accélérer la recherche. DB2 pourra le parcourir séquentiellement au lieu de devoir lire séquentiellement le fichier. On parle alors d'Index Scan. DB2 accèdera à la table uniquement pour les valeurs correctes.

Il est également possible d'obtenir un grand gain de performance en utilisant l'Index-Only Access. En quoi cela consiste-t-il ? Comme son nom l'indique, la requête n'accèdera qu'à l'index et non à la table pour obtenir les données. Pour permettre ce type d'accès, toutes les données demandées doivent évidemment être reprises dans l'index.

Prenons un exemple, vous avez une table de contacts reprenant une trentaine de champs. L'Id du contact est sa clé primaire. Vous avez un module de recherche qui liste alphabétiquement les contacts par nom et prénom avec une sélection sur base du nom. Cette requête est évidement couramment utilisée. Vous devez également récupérer l'Id pour pouvoir ensuite pointer sur le contact choisi. Afin d'optimiser la sélection et le tri, vous avez créé un index sur le nom et le prénom. Votre requête sera donc du type:

```
SELECT nom, prenom, id FROM contacts WHERE nom like
'DUP%' ORDER BY nom, prenom
```

Pensez à ajouter l'id dans votre index. La requête précédente se fera alors en mode Index-Only Access. Evidemment, c'est à faire pour des requêtes couramment utilisées et il ne s'agit pas de dupliquer votre table dans l'index.

3. Les bases de l'optimisation.

L'optimisation commence avec des règles simples et évidentes mais qui sont malgré tout couramment bafouées.

Afin de mieux comprendre les différences, créons une table et ses indexes.

création de la table

```
CREATE TABLE TEST.OPTIMIZE1
    ( ID BIGINT  NOT NULL  GENERATED ALWAYS AS IDENTITY
                                (START WITH 0, INCREMENT
                                    BY 1, NO CACHE )
       , W CHARACTER (7)  NOT NULL
     , X INTEGER  NOT NULL  WITH DEFAULT 0
     , Y VARCHAR (256)
     , Z VARCHAR (256)
        , CONSTRAINT pk_id PRIMARY KEY ( ID)  ) ;
```

création des index

```
CREATE  INDEX TEST.INDEX_X ON TEST.OPTIMIZE1 (X ASC, W
ASC)
    PCTFREE 10 MINPCTUSED 10 ALLOW REVERSE SCANS
       COLLECT  STATISTICS ;
CREATE  INDEX TEST.INDEX_W ON TEST.OPTIMIZE1 (W ASC)
    PCTFREE 10 MINPCTUSED 10 ALLOW REVERSE SCANS
       COLLECT  STATISTICS ;
```

La table contient 100.000 enregistrements dans lesquels la colonne w contient 10 valeurs différentes réparties uniformément et la colonne x contient des valeurs de 0 à 100. Les colonnes y et z contiennent une valeur fixe pour donner une taille plus grande à chaque enregistrement et pour avoir un contenu non inclus dans les index.

Afin d'obtenir rapidement un bon indicateur de performance et pour mieux comprendre ce qui se passe, nous pouvons utiliser l'outil d'analyse fourni dans l'éditeur de commande.

3.1. Ne lire que ce qui est utile.

```
SELECT w,y FROM test.optimize1
```

sera évidemment plus rapide que

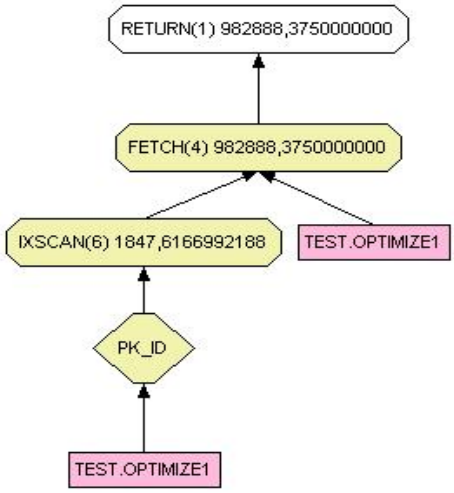
```
SELECT * FROM test.optimize1
```

En fait dans le premier cas et pour notre table de test, le temps de parcours et d'affichage est de 44 secondes et 5 centièmes alors que dans le deuxième cas le temps de parcours est de 3 minutes 30 secondes et 13 centièmes. Soit quasiment 5 fois plus lent. Pour partie la différence est due au temps d'affichage dans la boite de commande.

3.2. Utiliser les index

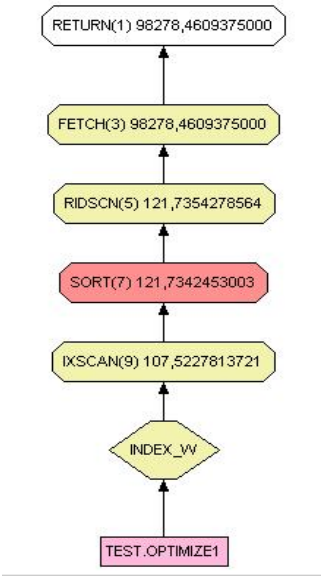
L'utilisation des index pour limiter les recherches est bien connu mais à titre d'exemple, voici les différences obtenues.
SELECT * FROM test.optimize1 WHERE w='GTX1000'

Sans l'index le chemin est le suivant:



Notez que DB2 lit en premier l'index de la clé primaire et ensuite l'enregistrement de la table. Le résultat sera trié sur l'ordre défini par la clé primaire.

Avec l'index "Index_W" créé:



Le temps de parcours est divisé par 10.

Attention : Ne multipliez pas les index à l'excès. Leurs maintenances a aussi un coût. Vous prétériterez ainsi le temps d'insertion dans la table, l'index devant lui aussi être modifié.

3.3. Like plutôt que substr

Imaginons une table ou w est le premier champ dans un des index.

```
SELECT w,x FROM test.optimize1 WHERE substr(w,1,2) =
'GT'
```

est annoncé moins rapide que

```
SELECT w,x FROM test.optimize1 WHERE w LIKE "GT%"
```

Selon la documentation, les fonctions scalaires ou de concaténations ainsi que le casting rendent l'utilisation de l'index sur les champs concernés impossible. Si vous le pouvez, écrivez votre sélection différemment pour les éviter. Toutefois, lors des tests, le chemin présenté contredit cette affirmation qui est donc à nuancer. Il semble en effet qu'au minimum sur la version DB2 8.2 UDB le moteur est capable d'utiliser l'index.

3.4. Veiller à la longueur des chaines

Lorsque votre commande SQL est le résultat d'un processus, vous pourriez vous retrouver avec une commande de ce type:

```
SELECT w,x FROM test.optimize1 WHERE w = 'GTX1000'
```

Rappelez-vous que w est défini comme une colonne caractère de longueur 7 et qu'elle est indexée. Pour les utilisateurs de DB2 z/os, il serait alors préférable de modifier le processus pour que la commande finale ait la forme suivante:

```
SELECT w,x FROM test.optimize1 WHERE w = 'GTX1000'
```

Pour les autres versions de DB2, il n'y a aucune différence de performance entre les deux formes.

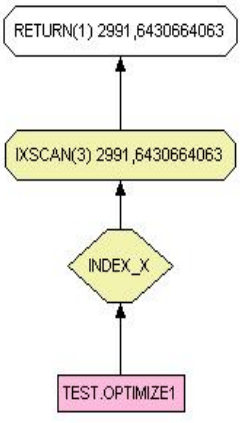
Le schéma de parcours obtenu dans les deux cas est le même que celui obtenu avec Substr et Like.

3.5. Eviter les opérations arithmétiques

Même si elles peuvent apporter plus de lisibilité au code parce qu'elles représentent une réalité business (ex: le salaire moins des frais forfaitaires doit être inférieur à un plafond) éviter les opérations arithmétiques dans les conditions.

```
SELECT w,x FROM test.optimize1 WHERE x - 10 < 50
```

La commande ci-dessus risque fort d'être pénalisante. Il est en effet possible que pour chaque enregistrement une opération arithmétique ait lieu. De plus, sur certaines versions DB2, l'index ne serait pas utilisé.



La syntaxe correcte est donc :

```
SELECT w,x FROM test.optimize1 WHERE x < 60
```

Toutefois, avec la version DB2 utilisée, le chemin et temps de parcours sont les mêmes. L'optimiseur prend donc en charge la modification mais prudence, ce n'est peut être pas le cas avec toutes les versions de DB2.

3.6. Eviter les conversions

Non seulement il faut éviter les fonctions de conversion pour les motifs indiqués précédemment mais il faut également se méfier des conversions implicites.

```
SELECT w,x FROM table WHERE w > 75000.00
```

Imaginez que w est une colonne de type entier et qu'il existe un index sur la colonne. Dans ce cas, il y aura conversion. Avec DB2 UDB pour Linux/Windows ainsi qu'avec DB2 pour iSeries, les performances ne seront pas réellement affectée mais avec la version pour z/os il en serait autrement.

Il est évident que vous n'entrerez jamais ce genre de commande à la main mais n'oubliez pas que le commande SQL peut elle même être le résultat d'un processus et là...

3.7. Préciser le type d'isolation adéquat

Par défaut, mais cela dépend de la configuration du serveur, vos requêtes vont généralement provoquer du locking. Ce mécanisme prend évidemment des ressources systèmes et mémoires qui peuvent pénaliser le serveur. Il existe plusieurs types de locking qui dépendent du niveau d'isolation. Choisissez celui qui est le plus adapté à votre requête et à vos besoins.

Le niveau d'isolation Uncommited Read est absolument à préciser pour toutes vos requêtes pour lesquels il n'y aura pas de modification des données. (Select en vue d'un rapport, d'une liste, d'une fonction de recherche,...)

```
SELECT x,y,z FROM table WHERE w = 'GTX2000' WITH UR
```

4. Conclusion

Comme nous avons pu le voir surtout pour des requêtes complexes qui sont les plus susceptible d'entrainer des problèmes de performance, il n'est pas évident de déterminer d'emblée la meilleure requête. En dehors des règles fondamentales, le filling et l'expérience du développeur seront ses meilleurs atouts. Dans les cas complexes, il est bon d'envisager plusieurs approches et de les tester avec un jeu de données vraisemblables aussi bien en quantité qu'en qualité.

L'optimisation est importante et trop souvent négligée. Généralement le développeur se contentera d'optimiser les requêtes longues mais n'oubliez pas que même une petite différence peut lors de la montée en charge avoir de grands effets. N'oubliez pas également que vous n'êtes vraisemblablement pas le seul à utiliser le serveur. Alors optimisez les requêtes complexes mais également les requêtes fréquemment utilisées.

Retrouvez l'article complet de Jean-Alain Baeyens avec encore d'autres astuces et un exemple concret en ligne : Lien30

Le type Datetime de SQL-Server

Avec cet article, vous allez découvrir comment fonctionne véritablement le type datetime de SQL-Server, son utilisation et ses mécanismes internes.

1. Introduction

DATETIME est le type SQL-Server pour stocker des valeurs composées d'une date et d'une heure (horodatage). Il correspond au type TIMESTAMP de la norme SQL. Disons-le tout de suite, SQL Server offre un type TIMESTAMP qui n'a rien à voir avec la norme SQL et constitue un simple marqueur de version de ligne. Ce type TIMESTAMP version SQL Server a d'ailleurs été rebaptisé ROWVERSION depuis la version 2000 de SQL Server. Le type DATETIME de SQL Server est donc destiné au stockage d'un combiné DATE + TEMPS dont la précision permet des expressions comme : 21/12/2006 22:16:43.666.
SQL Server ne dispose pas des types DATE et TIME séparés prévus par la norme.

Avant de s'aventurer à détailler ce type, commençons par un petit mystère avec l'utilisation du DATETIME :

```
CREATE TABLE T_TEST_DATE_TDT (
       TDT_ID       INT,
       TDT_DATE     DATETIME
)

INSERT INTO T_TEST_DATE_TDT VALUES (1, '20061223
23:59:59.99')
INSERT INTO T_TEST_DATE_TDT VALUES (2, '20061223
23:59:59.999')
INSERT INTO T_TEST_DATE_TDT VALUES (3, '20061224')
INSERT INTO T_TEST_DATE_TDT VALUES (4, '20061224
23:59:59')
INSERT INTO T_TEST_DATE_TDT VALUES (5, '20061224
23:59:59.9')
INSERT INTO T_TEST_DATE_TDT VALUES (6, '20061224
23:59:59.99')
INSERT INTO T_TEST_DATE_TDT VALUES (7, '20061224
23:59:59.999')
INSERT INTO T_TEST_DATE_TDT VALUES (8, '20061225')

-- selectionner les lignes pour la journée du 24
décembre...
SELECT *
FROM   T_TEST_DATE_TDT
WHERE  TDT_DATE BETWEEN '20061224 00:00:00.000'
                    AND '20061224 23:59:59.999'
```

Logiquement, à première vue, cette requête devrait nous renvoyer les lignes 3 à 7, mais ce n'est pas le cas, puisqu'elle nous retourne les lignes 2 à 8. Pourquoi ? C'est ce que vous allez comprendre à la lecture de cet article...

2. Précision

Vous allez maintenant comprendre le petit mystère du chapitre I. La limitation de l'entier de 4 octets ne permettait pas des calculs justes et précis à une milliseconde près. Il a été donc choisi de limiter la précision à 3 millisecondes afin d'obtenir de bonnes performances de traitement. Mais cette imprécision relative entraîne quelques effets de bord dont celui présenté au début de cet article. Ainsi la limite de temps d'une journée dans SQL Server n'est pas située à 23h 59m 59s et 999ms car nous serions déjà au lendemain du fait de l'imprécision, mais plus exactement à 23h 59m 59s et 997ms dernière valeur de temps exprimable directement dans un type DATETIME de SQL Server...

3. Stocker seulement une date ou du temps

SQL Server ne permet pas de stocker seulement une date ou un temps, il ne fournit que le type DATETIME qui permet de stocker les deux. Néanmoins, vous pouvez utiliser indifféremment un DATETIME pour une date avec heure à zéro, un INTEGER pour une date sans heure, un FLOAT ou un DECIMAL pour ne stocker que le temps...Exemple :

```
SELECT CAST('1900-01-04' AS DATETIME), CAST('10:00' AS
DATETIME)
```

La partie qui n'est pas renseignée sera tout simplement égale à zéro, c'est-à-dire pour la date 1/1/1900. Ne vous préoccupez donc que de la seule partie qui vous intéresse.

Les formats horaires reconnus par SQL Server sont les suivants :

- 14:30
- 14:30[:20:999]
- 14:30[:20.9]
- 4am
- 4 PM
- [0]4[:30:20:500]AM

Pour la date courante vous ne pouvez utiliser CURRENT_TIMESTAMP, la fonction qui renvoie la date/heure courante, car la date ne sera pas à zéro. Mais vous pouvez mettre la partie heure à zéro en faisant comme suit :

```
CAST(FLOOR(CAST(CURRENT_TIMESTAMP AS FLOAT)) AS
DATETIME
```

4. Recherche de dates

Prenons une table de dates :

```
CREATE TABLE T_DATES_TDT{
       TDT_ID INT,
       TDT_DATE DATETIME
}
INSERT INTO T_DATES (1, '2005-04-8 2:00:00.000')
INSERT INTO T_DATES (2, '2006-04-8 5:25:78.789')
INSERT INTO T_DATES (3, '2006-04-9 00:00:00.000')
INSERT INTO T_DATES (4, '2006-04-8 00:00:00.000')
```

Une idée malheureuse qui vient souvent à l'esprit pour rechercher toutes les dates du 8 avril 2006, consiste à faire :

```
SELECT * FROM T_DATES_TDT
WHERE TDT_DATE = '2006-04-8'
```

Hélas, vous n'allez récupérer que la ligne 4, car en l'absence de précision de l'heure, c'est à zéro heure que la partie temps opère. Une autre idée serait de faire :

```
SELECT * FROM T_DATES_TDT
WHERE TDT_DATE BETWEEN '2006-04-8' AND '2006-04-8
23:59:59.999'
```

Mais on retomberait dans la problématique de la précision et on obtiendrait aussi l'enregistrement 3.

Les bonnes solutions consistent à utiliser :
soit une fourchette non symétrique :

```
SELECT * FROM T_DATES_TDT
WHERE TDT_DATE >= '2006-04-8' AND TDT_DATE < '2006-04-
9'
```

soit une précision à 3 ms de la fourchette BETWEEN :

```
SELECT * FROM T_DATES_TDT
WHERE TDT_DATE BETWEEN '2006-04-8'
               AND '2006-04-8 23:59:59.997'
```

Une dernière solution consiste à transtyper la colonne date en forçant l'heure à zéro, comme ceci :

```
SELECT * FROM T_DATES_TDT
WHERE CAST(FLOOR(CAST(TDT_DATE AS FLOAT)) AS DATETIME)
= '2006-04-8'
```

Ce qui va aussi vous retourner les lignes 2 et 4. La fonction FLOOR permet de supprimer tout ce qui se trouve après la virgule pour n'avoir plus que la partie qui nous intéresse, c'est-à-dire la partie date.

5. Recherche par heure

Prenons la table :

```
CREATE TABLE T_TIMES{
       id INT,
       hour DATETIME
}
INSERT INTO T_TIMES (1, '2006-02-28 2:00:00.000')
INSERT INTO T_TIMES (2, '1900-01-01 5:58:32.823')
INSERT INTO T_TIMES (3, '1900-01-01 1:59:59.997')
INSERT INTO T_TIMES (4, '1900-01-01 2:00:00.000')
```

Là encore, cette table a été remplie de valeurs de date et de temps mélangés, ce qui est à proscrire.

Si vous voulez rechercher tous les temps à 2h, vous feriez :

```
SELECT * FROM T_TIMES
WHERE hour = '2:00:00'
```

Malheureusement, cela va vous retourner seulement l'enregistrement numéro 4.

Pour récupérer les bons résultats de manière sûre, il va falloir à nouveau employer le transtypage FLOAT :

```
SELECT * FROM T_TIMES
WHERE hour - CAST(FLOOR(CAST(hour AS float)) AS
datetime) = '10:00'
```

Cette fois, on va enlever la première partie de la date, pour ne conserver que l'heure et ainsi pouvoir la comparer exactement. Cela va nous retourner les lignes 1 et 4. Mais cela risque de se révéler lourd.

Une autre manière de faire, mais qui va nous retourner des résultats plus approximatifs et d'employer les opérateurs de comparaison :

```
SELECT * FROM T_TIMES
WHERE hour > '1:59' AND hour < '2:01'
```

Qui va nous retourner les enregistrements 3 et 4.

Si vous voulez vraiment récupérer toutes les heures à 2h, il va vous falloir faire quelque chose de plus lourd avec le transtypage vers FLOAT et les opérateurs de comparaison :

```
SELECT * FROM T_TIMES
       WHERE hour - CAST(FLOOR(CAST(hour AS float)) AS
datetime) > '01:59'
       AND hour - CAST(FLOOR(CAST(hour AS float)) AS
datetime) < '2:01'
```

Ce qui va cette fois vous retourner les enregistrements et 1,3 et 4.

Retrouvez l'article de Frédéric Brouard et Baptiste Wicht en entier et apprenez d'autres choses sur DATETIME : Lien31

Les derniers tutoriels et articles

Introduction au réseau

Avec cet article, vous allez apprendre les bases de la connaissance du réseau.

1. Avant-propos

Cet article va vous apprendre les bases de la connaissance du réseau. C'est à dire que vous allez découvrir quels sont les différents appareils qui opèrent dans un réseau et comment sont organisés les réseaux. Car il y a beaucoup de spécifications et de règles sur l'élaboration d'un réseau qu'il est très utile de connaître avant de se lancer dans le réseau.

2. Introduction

Avant de commencer à apprendre des notions purement réseau, nous allons nous attarder un moment sur les bases.

2.1. Les formats de données

Un ordinateur peut comprendre une donnée seulement si celle-ci est binaire. Une donnée binaire est une donnée codée en base 2.
Un autre format de donnée aussi beaucoup utilisé est l'hexadécimal, il est codé en base 16.

2.1.1. Les termes de mesure de données

- Bit(b) : C'est la plus petite unité de mesure possible, un bit peut être 1 ou 0, c'est le format binaire avec lequel travaille le CPU.
- Byte(B) ou Octet(o) : C'est un groupe de 8 bits.
- Kilo(k) : Représente 1000(1024). Exemple 2kB = 2048 Byte = 16384 bits
- Méga(M) : Représente 1'000'000(1'048'576)
- Giga(G) : Représente 1'000'000'000
- Ps : par seconde, unité de mesure de vitesse d'un réseau par exemple : kbps

2.2. Terminologie de base de réseau

- Network Interface Card(NIC) : Carte réseau
- Media : C'est le type de câble
- Protocol : C'est une série de règle qui définit comment le pc va communiquer a travers le réseau, il existe beaucoup de types de protocoles, des protocoles de routage, des protocoles internet, …
- IOS (Internetwork Operation System) : C'est le logiciel qui est dans l'élément de réseau, en quelque sorte son OS.
- LAN (Local Area Network) : C'est un petit réseau, qui est confiné entre de petites barrières géographiques, cela peut être une chambre, un bâtiment, ou éventuellement plus grand.
- MAN (Metropolitan Area Network) : C'est un réseau plus grand que le LAN, il couvre environ une ville entière
- WAN (Wide Area Network) : C'est un réseau gigantesque, qui peut s'étendre sur plusieurs pays. Internet en est un exemple.

2.3. Le modèle OSI (Open System Interconnexion)

Ce modèle a été créé par l'organisme ISO. C'est une norme internationale. Elle est implémentée dans presque tous les réseaux et la plupart des protocoles en sont dérivés. Les entreprises se sont rendu comptes que si tout le monde se basait sur les mêmes spécifications, la communication entre réseau serait énormément améliorée. Ce modèle est formé de 7 couches ayant chacune des applications bien distinctes.

2.3.1. Avantages d'OSI et de la division par couches

Cela réduit la complexité, puisque cela subdivise la communication en plus petites couches
Cela standardise bien sûr les interfaces
Cela permet un meilleur développement et une meilleure évolution, car il suffit d'interagir sur la couche qui doit être modifiée.

2.3.2. Les couches OSI

Je vais maintenant vous présenter les différentes couches qui forment la norme OSI.

2.3.2.1. La couche 7 : Application

La couche application est responsable de la communication entre le réseau et les applications. Elle offre le service réseau à l'application qui le demande. Elle est différente des autres couches, car elle n'offre pas de service aux autres couches

2.3.2.2. La couche 6 : Présentation

Cette couche s'occupe surtout de traduire les données pour que les 2 systèmes puissent communiquer entre eux et se comprendre. Par exemple si un envoie de l'ASCII et l'autre du DCB, la couche va s'occuper de traduire dans les 2 sens.

2.3.2.3. La couche 5 : Session

Cette session établit, gère et termines les communications entre 2 systèmes. Elle s'occupe aussi de synchroniser les dialogues entre les hosts. Elle assure la communication et la gestion des paquets entre 2 stations.

2.3.2.4. La couche 4 : Transport

Cette couche divise les données de l'envoyeur, puis les rassemble chez le récepteur. La couche transport assure la fiabilité et la régulation du transfert de données..C'est la couche tampon en quelque sorte, car elle se trouve entre les couches purement réseau et les couches qui eux se réfèrent plus aux applications.

2.3.2.5. La couche 3 : Réseau

Cette couche gère la connectivité entre 2 systèmes qui peuvent être localisés dans différents endroits géographiques et dans

différents réseaux. La couche liaison de données assure un transit fiable des données sur une liaison physique. Elle se réfère aux adresses réseaux donc IP

2.3.2.6. La couche 2 : Liaison de données

Cette couche définit comment les données sont formatées et comment on accède au réseau. Elle est responsable de " dire " comment un appareil correspond avec un autre alors qu'ils sont sur différents réseaux et médias. Elle se réfère à l'adressage physique donc aux adresses MAC.

2.3.2.6. La couche 1 : Physique

La couche physique est la couche de bas niveau, c'est la couche la plus basique du modèle, elle contient toutes les spécifications électriques, mécaniques pour l'activation, la maintenance entre le lien physique et le système. Par exemple, les distances de transmission, le voltage, les connecteurs physiques, le type de média.

2.4. La communication Peer-to-Peer

La communication Peer-to-Peer (P2P) est un modèle de réseau informatique dans lequel tous les élments n'ont pas seulement un rôle (client ou serveur), mais peuvent fonctionner dans les 2 rôles.

2.5. Le protocole TCP/IP

Le protocole TCP est basé sur les couches OSI, mais il n'en a lui-même que 4.

Je vais maintenant vous présenter les couches du protocole TCP/IP. Vous ne serez néanmois pas trop perturbés, car les couches ressemblent beaucoup à celles du modèle OSI.

2.5.1. La couche 4 : Application

C'est la couche de haut niveau, elle correspond directement avec l'utilisateur, elle englobe les couches OSI d'application, de présentation et de session. Elle s'assure que les données soient correctement "empaquetées" pour qu'elles soient lisibles par la couche suivante.

2.5.2. La couche 3 : Transport

Cette couche est sensiblement ressemblante à la couche transport du modèle OSI.

2.5.3. La couche 2 : Internet

Cette couche doit s'assurer que les données envoyées arrivent correctement à destination.

2.5.4. La couche 1 : Network Access

Cette couche est assez confuse. Elle inclut tous les protocoles LAN et WAN et tous les détails que les couches OSI liaisons de données et physique fournissaient.

3. Généralités réseaux

Dans un réseau, il faut différencier 2 types d'utilisateurs, ceux qui travaillent directement depuis le bâtiment principal, donc qui auront un accès direct au réseau et ceux qui eux, travaillent depuis ailleurs ou alors sont mobiles.
Les premiers auront accès directement au réseau, donc à une connexion haute vitesse. Par contre pour les suivants, soit ils travaillent dans des bureaux de l'entreprise, donc dans un sous-réseau du réseau et donc une connexion plus ou moins rapide ; soit avec les gens mobiles ou travaillant à la maison, on aura recours à une connexion en dialup (par modem, ex : VPN) pour les connecter sur le réseau.

3.1. Le modèle réseau

Pour faciliter la compréhension d'un réseau, CISCO a mis au point un modèle hiérarchique en couches. Chaque couche a un but précis, et chacune des couches communiquent ensemble.

3.1.1. La couche d'accès

Cette couche est le point d'accès dans le réseau, c'est par la qu'arrivent toutes les connexions. Elle est aussi appelée la couche bureau.

3.1.2. La couche de distribution

Cette couche remplit les fonctions de routage, filtrage et gère les accès depuis le WAN. Elle s'occupe aussi de faire communiquer des réseaux dans différentes topologies. Elle va trouver le meilleur chemin pour la requête et ensuite va transmettre cette requête à la couche de " coeur ". Elle est aussi appelée la couche Workgroups.

3.1.3. La couche "coeur"

Cette couche va s'occuper de switcher le trafic vers le bon service, de la manière la meilleure et la plus rapide qui soit. Ensuite il va répondre à la couche de distribution qui si c'est un accès WAN va directement rendre réponse, soit passer la main à la couche d'accès. On l'appelle aussi la couche backbone.

3.2. Les topologies

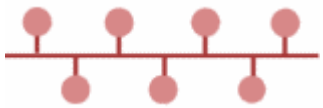
On peut différencier 2 types de topologies :

- Topologie physique : C'est l'emplacement exact des appareils de réseau et comment ils sont interconnectés.
- Topologie logique : C'est comment chaque point du réseau et connecté à un autre point du réseau

Les topologies logique et physique d'un réseau peuvent tout à fait être les mêmes. Mais cela peut aussi être tout à fait différent.
La topologie est définie dans le couche physique et la couche de liaisons de données du modèle OSI.

3.2.1. La topologie en bus

Dans cette topologie, chaque élément est relié au même câble. Le câble se termine par un " bouchon ", pour absorber le signal à la fin du parcours, pour ne pas causer d'erreurs dans le système. Cette structure est très vulnérable, car si un seuls des hôtes tombe, tout le réseau tombe.

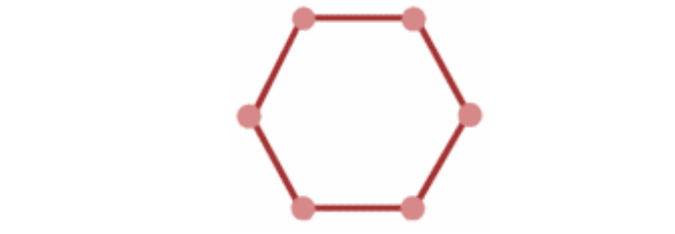


3.2.2. La topologie en étoile

Ce type de réseau est très employé, car efficace et peu couteux. Tous les éléments sont reliés à un point central. Si un hôte tombe, seul celui-ci tombe, par contre si un élément central tombe, tout le réseau tombe.

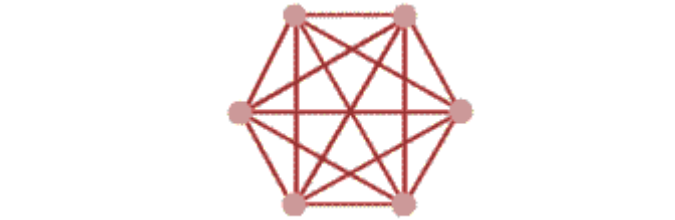
3.2.3. La topologie en anneau

Une topologie en anneau ressemble assez à une topologie en bus, sauf qu'elle n'a pas de fin ni de début, elle forme une boucle. Quand un paquet est envoyé, il parcourt la boucle jusqu'à ce qu'il trouve le destinataire. Il existe soit la topologie en anneau simple soit la topologie en double boucle(FDDI), qui permet une redondance et qui comme son nom l'indique est formé de deux anneaux.



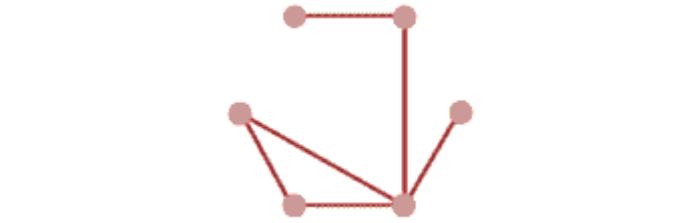
3.2.4. La topologie point par point

Cette topologie consiste à relier chaque point l'un à l'autre, ce qui implique un cout extrêmement élevé. Cette topologie n'est pas utilisée en pratique, mais c'est sur une topologie pareille qu'est basée internet. C'est la technologie la plus sûre.



3.2.5. La topologie quelconque

Cette topologie est malheureusement souvent utilisée. Elle ne suit aucune règle précise et de ce fait n'est pas très fiable ni efficace... Elle apparaît souvent lorsque l'on connecte ensemble des sous-réseaux.



3.3. Couche physique : détails

Cette couche définit le type de média, le type de connecteurs et le type de signal. Elle spécifie le voltage, la vitesse de transfert, la distance de transmission maximale et les connecteurs physiques.

3.3.1. Périphériques

Les périphériques de la couche 1 sont les appareils les plus basiques du réseau :

- Repeater : Cet appareil permet d'étendre l'utilisation d'un média en régénérant le signal, et ainsi, lui permettre d'atteindre une plus longue distance.
- Hub : Il a la même fonction que le repeater, à la différence près qu'il possède plusieurs ports, donc il divise le signal en plusieurs parties, tout en le régénérant. Dès qu'il reçoit un paquet sur un port, il l'envoie automatiquement sur tous les autres ports. Il est déconseillé maintenant d'utiliser des hubs, il faut leur préférer les switchs, car avec un hub la bande passante est divisée par le nombre de machines connectés et en plus le domaine de collision est le même pour toutes les machines interconnectées.

3.4. Couche liaison de données : détails

Alors que la couche une ne gérait aucun adressage, cette couche gère l'adressage via les adresses physiques (MAC) des machines.

3.4.1. L'adresse MAC

L'adresse mac est une adresse de 48 bits de 12 chiffres hexadécimaux. Cette adresse est un identifiant physique, stockée dans la mémoire de la carte réseau. Elle identifie donc l'interface réseau de la machine.

3.4.2. Périphériques

Les périphériques de la couche 2 sont déjà moins basiques que ceux de la couche 1, ils permettent déjà de l'adressage et sont plus intelligents :

- Bridge : C'est une sorte de hub, mais en plus intelligent. Il crée plusieurs domaines de collisions, permet le passage de paquets entre plusieurs segments LAN, maintient à jour une table d'adresses MAC.
- Switch : Aussi dans la couche 2, car il emploie aussi les adresses MAC. En fait, il est formé de plusieurs hubs. Un switch peut avoir une adresse MAC, qui va servir au routeur pour le rediriger. Chaque port du switch a son propre domaine de collision. Il a aussi sa propre table d'adresses MAC.

Le but de cette couche est surtout de réduire les collisions.

3.5. Couche réseau : détails

Cette couche gère un adressage autre que l'adressage de la couche 2, c'est un adressage réseau et non plus physique, aussi appelées adresses logiques ; il s'agit des adresses IP.

3.5.1. Adresse IP

L'adresse IP est une adresse de 32 bits, répartis en 4 fois 8 bits (octets). Cette adresse est un identifiant réseau. On peut ensuite la diviser en 2 portions : la portion du réseau et la portion hôte. La première identifie le réseau sur lequel est la machine et la deuxième identifie la machine en elle-même. Pour identifier ces 2 parties, chaque adresse est liée à un masque de sous-réseau. Ce qui permet de définir sur quel réseau elle se trouve.

3.5.2. Périphériques

Le périphérique principale qui est sur cette couche, est le routeur. Il est utilisé pour relier et faire communiquer ensemble des réseaux différents. Pour cela, il utilise une table de routage, dans laquelle il va stocker des informations importantes aidant au routage entre différents réseaux, sur quelle interface est ce réseau, à quelle distance (nombre de sauts) est ce réseau, sur quelle plage

d'adresse est ce réseau. Il peut aussi s'occuper de router 2 réseaux sur 2 protocoles différents.

3.6. Couche Transport : détails

Cette couche gère les transmissions entre les protocoles de la couche réseau (IP, IPX) et les protocoles propres à la couche transport (TCP, SPX).

3.6.1. Périphériques

Les seuls appareils qui appartiennent à cette couche, sont les appareils multicouches, tels qu'un switch qui peut utiliser aussi les adresses IP

4. Types de réseau

4.1. Lan : Détails

Le réseau Lan est le réseau le plus employé de nos jours. Depuis qu'internet a été crée, c'est-à-dire depuis environ 20 ans, le LAN a beaucoup évolué pour correspondre aux nouvelles technologies. Il existe différentes technologies de LAN : Ethernet, Fast Ethernet et Gigabit Ethernet. Un Lan est confiné dans un petit endroit, il ne couvre pas de longues distances. Le Lan s'étend sur les couches Physique et liaison de données du modèle OSI.

4.1.1. Ethernet (IEEE 802.3)

C'est le type de Lan le plus employé. Ce standard, développé par IEEE, est basé sur un processus appelé " carrier sense multiple acces collision detect " (CSMA/CD). Ce standard est aujourd'hui tout simplement appelé Ethernet.
IEEE divise la couche de liaison de données en 2 : LLC (couche du haut vers réseau) et MAC (couche du bas vers physique). Les signaux Ethernets sont transmis à chaque station en utilisant une série de règles pour savoir quelle station peut " parler " et quand.
Avant de transmettre un signal, le pc commence par écouter le réseau et ensuite, si le réseau est prêt, il envoie ses données. Ensuite il attend à nouveaux un temps et continue à envoyer. Ainsi, aucun pc n'a de priorités sur les autres. Mais il est possible que 2 pc écoutent en même temps et voient en même temps que le réseau est libre, donc envoient simultanément un paquet, ce qui crée une collision ; les données du paquet sont perdus et les pc doivent donc recommencer l'envoi.

L'Ethernet a une vitesse de 10 Mbps.

4.1.2. Fast Ethernet (IEEE 802.3u)

Ce standard augmente la vitesse de transmission de 10 à 100 Mbps. Les changements sont minimaux, car il n'y pas besoin de changement d'appllication ni de protocoles

4.1.2.1. Spécifications

Protocole	Vitesse	Média
100 Base-T	100	Paire torsadée
100 Base-F	100	Mono ou Multimode fibre
100 Base-X	100	Fibre ou cuivre
100 Base-FX	100	Multimode fibre
100 Base-T4	100	Cuivre UTP 4 paires Category 3-5
100 Base-TX	100	2 Paires cuivre Category 5

4.1.3. Gigabit Ethernet (IEEE 802.3z)

Ce standard, quand à lui, permet une vitesse de transmission de 1000Mbps. Il est utilisé avec des fibres ou des câbles à paire torsadées. C'est devenu un standard très utilisé pour les connexions haute vitesse, par exemple sur les backbones. Il est aussi utilisé pour la connexion de plusieurs endroits ensemble.

4.1.3.1. Spécifications

Protocole	Vitesse	Média
1000Base-LX	1000	Mono ou Multimode fibre
1000Base-SX	1000	Multimode fibre
1000Base-CX	1000	2 paires Cuivres STP
1000Base-T	1000	Category 5 cuivre

4.1.4. 10 Gigabit Ethernet (IEEE 802.3ae)

Ce standard est le plus rapide, il permet une vitesse de transmission de 10GBps. Il est utilisé soit avec de la fibre optique soit avec des câbles à paires torsadées. Ce standard est surtout utilisé pour des accès à des bases de données de gros volume ou moins souvant pour le backbone.

4.2. Wan : Détails

Ce type de réseau couvre une région géographique plus ou moins grande. C'est une interconnexion de Lan d'habitude. Il emploie les trois premières couches du modèle OSI.

Les périphériques employés sur ce réseau sont les suivants :

- les routeurs qui font la liaison entre le LAN et le Wan
- Les switchs WAN, qui redistribuent le réseau Wan
- Les modems qui font une liaison entre 2 Lan par le Wan, en passant par le réseau téléphonique…

Le principal désavantage du Wan est sa vitesse, car il ne s'agit pas de fibre optique ni de paire torsadée, mais il s'agit de passer sur le réseau public, donc par un provider pour relier 2 Lan très éloigné. Ce type de réseau est de moins en moins employé, on préfère maintenant installer des lignes fibre optiques donc connexion Lan sans plus passer sur un réseau public. La meilleure des alternatives au Wan est le MAN.

4.3. Man : Détails

Un Man est une sorte de Wan, sauf que c'est à haute vitesse et qu'il passe par les médias propres au réseau et non plus par un provider externe. C'est très employé pour la connexion de plusieurs lieux de travail d'une entreprise dans une même ville ou même plus loin qu'une ville, dans un même canton. Même si il est sur que c'est plus cher d'interconnecter soi même que de passer par un provider, c'est beaucoup plus rapide et fiable et cela peut même se révéler rentable à plus long terme.

4.4. Storage area network (SAN)

Un réseau San est utilisé pour transférer des données des serveurs jusqu'à des ressources de stockage. On utilise d'habitude de la fibre pour ces connexions car on leur préfère leur grande rapidité. Ce réseau est isolé du reste du réseau, ce qui permet une meilleure configuration et sécurité de celui-ci. Le cout d'installation d'un San est très élevé justement à cause du fait de son isolation. Mais par contre on n'a plus besoin de se préoccuper de l'espace de stockage de chaque serveur, puisque l'espace de stockage forme un tout.

4.5. Content Network (CN)

C'est un réseau qui s'occupe d'accélérer l'envoi de données entre les services réseaux (Web, Streaming, applications, etc...). Il optimise l'envoi des informations vers les demandeurs. Il divise en plusieurs parties les technologies des services pour permettre une meilleure distribution de ceux-ci. Il va par exemple choisir le meilleur serveur pour telle ou telle ressource, il va aussi choisir le bon site pour le téléchargement d'une information et va s'occuper de garder " au frais " des ressources statiques ou en streaming pour qu'elles soient disponibles plus rapidement.

4.6. Virtual private Network (VPN)

Un réseau VPN permet de créer un tunnel par exemple depuis votre maison jusqu'à l'entreprise via l'internet. Par ce tunnel vous serez virtuellement dans le réseau de votre entreprise tout en étant physiquement chez vous.
Pour faire une liaison, il faut passer par le réseau public.

Il existe plusieurs types de VPN :

- Access VPN : C'est le VPN qui lie un travailleur mobile ou une personne travaillant chez elle jusqu'à au réseau de l'entreprise. La connexion se fait habituellement par modem, ISDN, dialup, DSL, ...
- Intranet VPN : C'est le réseau qui lie des bureau régionaux jusqu'au réseau de l'entreprise. Il ne permet l'acces au réseau qu'aux membres de l'entreprise
- Extranet VPN : C'est le réseau utilisé pour lier les entreprises externes sur le réseau de l'entreprise. L'accès est permis pour un groupe de gens bien défini.

5. La couche physique

Un média est un support par lequel vont passer des données.

5.1. Câble à paire torsadée

Un câble à paire torsadée est un câble dans lequel passe des fils de cuivres. C'est le média le plus employé de nos jours. C'est un câble utilisé pour câbler des courtes distances. Ces câbles ne peuvent couvrir qu'au maximum 100 mètres.

Un câble à paire torsadée est composé de plusieurs éléments :

- des brins de cuivre entrelacées
- d'une enveloppe isolante autour

Il en existe plusieurs catégories :

- Catégorie 1 : utilisé pour les communications téléphoniques, inutilisables pour le transfert de données
- Catégorie 2 : Transmission de données à 4 Mbps
- Catégorie 3 : Transmission de données à 10 Mbps
- Catégorie 4 : utilisé dans les réseaux Token Ring, transmission à 16 Mbps
- Catégorie 5 : Transmission de données à 100 Mbps
- Catégorie 5e : Transmission de données à 1 Gbps
- Catégorie 6 : Consiste en 4 paires de 24 gauges de cuivre, 1 Gbps
- Catégorie 7 : Transmission de données à 10 Gbps

5.1.1. UTP

C'est un câble à paire torsadée tout simple, sans aucun blindage. Il est fait de quatre paires de brins. Il est très utilisé pour les téléphones car il est plus petit qu'un câble STP.

5.1.2. STP

L'ensemble des paires torsadées est entourée d'un blindage. Il est plus utilisé dans les réseaux Ethernet que l'UTP, car il permet de réduire les effets électromagnétiques sur le câble grâce à son blindage. Il existe encore une autre variante, le SSTP, qui rajoute un blindage supplémentaire sur chaque paire.

Le blindage permet de réduire les interférences, donc le mélange de signaux électriques et il permet des transferts à des débits plus importants et sur des distances plus grandes.

5.2. Câble coaxial

Un câble coaxial consiste en un conducteur de cuivre isolé dans une insolation. Autour de cette isolation, il y a un bouclier en cuivre qui aide à réduire les interférences. Ce câble peut supporter des vitesses de 10 ou 100 Mbps et n'est pas très couteux, bien que plus cher que l'UTP. Par contre, il peut couvrir des distances plus longues que l'UTP, jusqu'à 500 mètres.

5.3. Câble fibre optique

Un câble fibre optique est fait de 2 fibres, chacune est blindé et ensuite mises dans un " tube " en plastique qui vient ensuite se coller au tube de la deuxième fibre. Il existe une multitude de connecteurs pour la fibre optique, que vous choisirez surtout en fonction de leur taille, de leur robustesse et de leur prix. Ce média permet des liaisons longue voire très longue distance à des débits élevés.

Il existe 2 types de fibre optique :

- Monomode : Utilisée pour les très longues distances, son prix est très élevé. Le laser circule tout droit.
- Multimode : Utilisée plutôt en local pour les connexions appareils réseaux - serveurs. Elle coute moins cher que de la Multimode mais comme toute fibre, son prix reste élevé. Le laser circule à l'intérieur de la fibre en rebondissant sur les cotés.

La fibre optique est le type de câblage le plus avancé technologiquement. Un des plus gros avantages est le fait qu'il est insensible aux perturbations électromagnétiques, puisqu'il transporte de la lumière. De plus, il est aussi insensible aux écoutes clandestines puisque pour l'écouter, il faudrait se couper directement dessus, ce qui bien sur, couperait la communication. Un câble optique peut négocier des transferts allant jusqu'à 200 Gigabit/s. Et là on parle de distances dépassant plusieurs kilomètres, ce qu'aucun câble de cuivre ne permet de faire. Aujourd'hui, c'est la meilleure solution pour des grandes distances et des gros transferts. Mais il faut quand même de gros moyens pour mettre en place une solution fibre optique.

5.4. Communications Wireless

La communication par Wireless utilise des fréquences radio ou infrarouge pour communiquer entre plusieurs appareils dans un LAN. Les signaux Wireless sont des signaux électromagnétiques qui peuvent conduire des données. Plus on augmente la fréquence des signaux, plus la distance sur laquelle l'onde est propagée diminue.
Les communications par Wireless ont pas mal d'avantages : L'accès à internet par des natels, transfert de données entre 2 périphériques sans fil, permet de diminuer le nombre de câbles, augmente la mobilité, permet la connexion de souris et clavier, …

Retrouvez la suite de l'article de Baptiste Wicht en ligne : Lien33

Le publipostage Word-Excel

Ce document décrit la fonction de publipostage Word. Vous y trouverez des informations sur les principales options disponibles.

Les exemples proposés ont été testés avec Office XP.

1. Introduction

Le publipostage (aussi appelé fusion ou mailing) permet d'envoyer un courrier à un ensemble de destinataires. Le publipostage utilise un modèle (le document principal Word) et une base de données constituée de champs (Nom, Description, Date, ...toute information de votre choix...) et d'enregistrements (La liste des destinataires).
La technique consiste à fusionner le document Word avec chacun des enregistrements.
La destination de la fusion peut être une lettre, une enveloppe, une étiquette, un fichier ou un message électronique.

2. Créer un document de publipostage

Ce chapitre décrit les étapes nécessaires à la mise en place d'une fusion. La description est basée sur le pas à pas de l'assistant de publipostage Word, complétée par des commentaires personnels.

2.1. La base de données

Ce tutoriel utilise un classeur Excel comme base de données mais il est aussi possible d'utiliser un tableau Word ou une table Access.

	A	B	C	D	E
1	leNom	Description	laDate	Montant	Nombre
2	Nom01	Description01	26/05/1965	150,00 €	1
3	Nom02	Description02	12/08/2005	90,75 €	5
4	Nom03	Description03	15/02/2006	17,80 €	3
5	Nom04	Description04	16/02/2006	0,96 €	6
6	Nom05	Description05	21/03/2001	18,00 €	3
7	Nom06	Description06	18/09/2003	4,76 €	2

Il est important de bien structurer la base de données.

Lorsque vous créez le nom des champs, respectez les conseils suivants:

- Nom le plus court possible
- Pas d'espace
- Pas d'accent
- Pas de caractères spéciaux

Evitez les vides entre les différentes colonnes.
Evitez les lignes vides entre les différent enregistrements.

Quand la base de données Excel est créée, sauvegardez et fermez votre classeur.

2.2. 1ere étape: Sélectionner le type de document

Ouvrez un nouveau document Word
Utilisez le Menu Outils
Sélectionnez l'option "Lettres et Publipostage"
Puis "Assistant de Fusion et Publipostage"
Le volet Office apparaît à droite de l'écran. Suivez les différentes étapes proposées par l'assistant.

Sélectionnez le type de document (par exemple "Lettres") dans la liste de choix.

2.3. 2eme étape: Sélectionner le type de document de base

Sélectionnez l'option "Utiliser le document actuel" pour que le fichier Word actif devienne le document principal de fusion.

2.4. 3eme à 5eme étape: Sélectionner la base de données et appliquer une mise en page

Sélectionnez l'option "Utilisation d'une liste existante", puis cliquez sur le bouton Parcourir.

Remarques:
Il est aussi possible de créer un publipostage à partir de la liste des contacts Outlook.
L'option "Saisie d'une nouvelle liste" permet de créer une nouvelle base de données préformatée (style carnet d'adresses). Cette base sera sauvegardée au format mdb après que vous ayez saisi vos données.

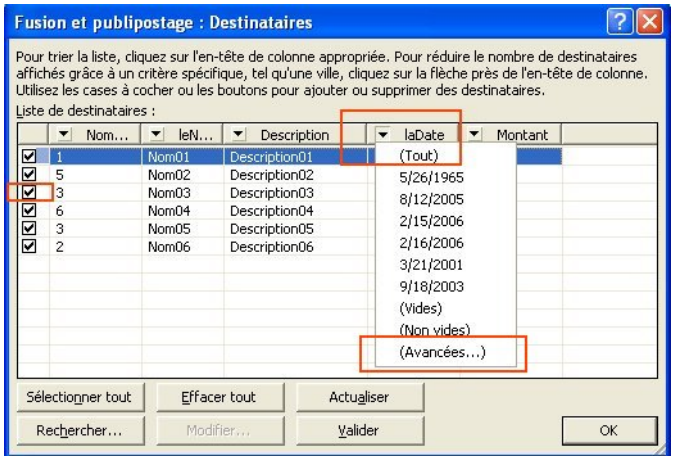
Recherchez et sélectionnez la source de données (le classeur Excel) contenant les informations à fusionner.

La liste des tables s'affiche dans une nouvelle boîte de dialogue.
Chaque feuille (ou plage nommée) est considérée comme une table.
Consultez le chapitre IV pour obtenir plus de détails sur les types de connexion.

Si le classeur contient plusieurs onglets, sélectionnez celui qui vous interesse.
Cliquez sur OK pour valider.

Si vous renommez ou déplacez ultérieurement la base de données, vous devrez recréer le lien dans le document principal.

La boîte de dialogue suivante permet de trier et filtrer les destinataires.
Chaque colonne (champ) contient un filtre avancé qui permet d'appliquer 5 critères de filtre ET / OU, et 3 critères de tri.



Cliquez sur OK.

Vous pouvez mettre en page votre document Word.
Préparez le texte que vous souhaitez envoyer. Vous ajouterez ensuite des champs qui permettront d'afficher les informations contenues dans la base de données.
Pour insérer un champ dans le document Word, cliquez sur le bouton "Insérez les champs de fusion", dans la barre de menu "Fusion et Publipostage".

Si la barre de menu "Fusion et Publipostage n'apparaît pas à l'écran:
Utilisez le Menu Affichage
Sélectionnez l'option "Barre d'outils"
Sélectionnez "Fusion et Publipostage"

La fenêtre affiche la liste des champs contenus dans la base de données Excel. Choisissez un des Champs (par exemple "leNom").

Ensuite Cliquez sur le bouton "Insérer".
Puis sur le bouton "Fermer"

Le champ est inséré dans le document Word à l'emplacement du curseur.
Allez jusqu'à la 5eme étape de l'assistant pour visualiser le résultat.

L'assistant propose un outil pour faire défiler les enregistrements et avoir un aperçu des champs insérés.

Vous pouvez aussi utiliser les boutons de la barre de menu "Fusion et Publipostage"

Il existe une 2eme méthode pour insérer un champ dans le document Word:
Les 3 premières étapes de l'assistant doivent être préalablement réalisées,
Puis utilisez le Menu Insertion.
Sélectionnez l'option "Champ".

Sélectionnez MergeField (champFusion) dans la liste.
Saisissez un des noms de champ contenu dans la base de données, par exemple "laDate" (L'entête de la colonne C dans le classeur Excel).
Cliquez sur OK.
Le champ "laDate" est ajouté dans le document Word.

2.5. L'étape 6: Fusionner

La préparation est terminée. Vous pouvez lancer la fusion.

Remarque:
Pour ne pas gaspiller du papier inutilement, utilisez l'aperçu (étape 5 de l'assistant) pour vérifier que le résultat correspond à votre attente. Revenez sur l'étape 4 pour modifier votre document si nécessaire.

Ensuite, cliquez sur le bouton "Imprimer".

Vous pouvez spécifier l'impression de tout ou partie des enregistrements lors de cette dernière étape.

3. Le format des champs

3.1. Informations sur la mise en forme

Si vous utilisez Office XP ou une version ultérieure, il est parfois nécessaire de remettre en forme les champs car le résultat ne correspond pas aux données contenues dans la base: Par exemple les dates qui s'affichent au format MM/JJ/AAAA au lieu de JJ/MM/AAAA.
Par défaut, Word utilise la connexion OLE DB pour la fusion, et les dates sont donc gérées en anglais.

Vous devrez modifier manuellement tous les champs qui posent problème.
Ci-dessous quelques exemples de mise en forme à appliquer en fonction des types de données.

Format Décimal:
```
MERGEFIELD leChamp \# "#,00"
MERGEFIELD leChamp \# "### ### ###,##"
```

Format Date:
```
MERGEFIELD laDate \@ "dd/MM/yyyy"
MERGEFIELD laDate \@ "dd dddd MMMM yyyy"
```

Format Monétaire:
```
MERGEFIELD Montant \# "# ###,00 €"
```

Format numéro de Téléphone:
```
MERGEFIELD \# "00' '00' '00' '00' '00"
```

Plus d'informations sur le site Microsoft : Lien34

3.2. 1ere méthode pour modifier un champ

Dans le document Word, faites un clic droit sur un des champs afin d'afficher le menu contextuel.

Sélectionnez "Basculer les codes de champs".
Le résultat est remplacé par les propriétés du champ.

Vous pouvez modifier le format directement dans le champ. Un exemple pour modifier le format Date:

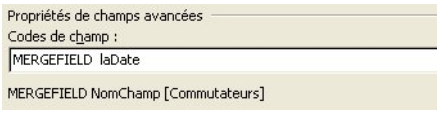
Ensuite, pour revenir en mode normal, refaites un clic droit et sélectionnez l'option "Basculer les codes de champs".

Attention : La mise à jour n'est pas effectuée automatiquement. Vous devez l'activer en utilisant la procédure suivante:
Menu Edition / Sélectionnez tout (Ctrl + A)
Puis appuyez sur la touche F9
Le format du champ est maintenant mis à jour.

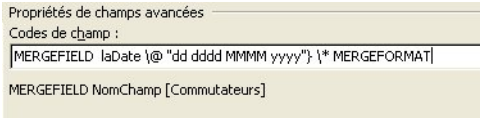
3.2. 2eme méthode pour modifier un champ

Faites un clic droit sur un champ.
Sélectionnez "Modification du champ" dans le menu contextuel.
Cliquez sur le bouton "Code de champ", en bas à gauche dans la boite de dialogue.

Vous pouvez ensuite visualiser les propriétés du champ.



Comme pour la premiere méthode, vous pouvez modifier le format du champ.



Cliquez sur OK pour valider.

4. Ne pas utiliser la connexion OLE DB par défaut

Par défaut, Word utilise la connexion **OLE DB** pour la fusion.
La procédure suivante modifie les paramètres pour que le choix du mode de connexion soit possible lors de l'ouverture.

Utilisez le Menu Outils.
Sélectionnez "Options"
puis l'onglet "Général"
Cochez l'option "Confirmation des conversions lors de l'ouverture".
Cliquez sur OK pour valider.

Désormais, lorsque vous sélectionnez une base de données, la boite de dialogue "Confirmer la source de données" s'affiche et vous permet de choisir votre mode de connexion.
Voici une description des différents types de connexions et les problèmes possibles lors de l'utilisation des bases de données Excel:

Convertisseur Excel:
Permet de choisir la feuille de calcul, mais il peut y avoir des problèmes avec les calculs et les formats numériques.

OLE:
Permet de choisir la feuille de calcul et des plages de cellules, mais il peut y avoir des problèmes avec les calculs, les formats numériques et les formats Date.

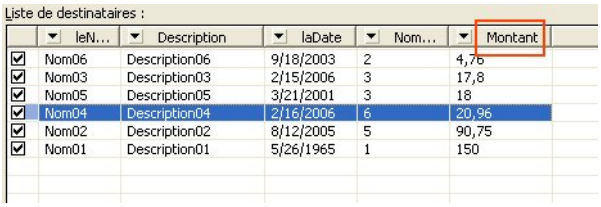
DDE:
Seule la 1ere feuille du classeur peut etre utilisée comme base de données. Les calculs, les formats numériques et les formats Date sont respectés. Utilisez ce type de connexion lorsque vous souhaitez utiliser des champs images (insertPicture).

ODBC:
Fonctionne uniquement avec des plages de cellules. Il peut y d'avoir des problèmes de formats numériques.

5. Afficher les lettres types triées par ordre croissant

L'outil de publipostage Word dispose d'une option de tri sur chaque champ de la base de données, permettant ainsi de définir l'ordre des lettres.
A l'étape 3 de l'assistant, cliquez sur l'option "Modifier la liste des destinataires".

Cliquez par exemple sur l'entête "Montant" pour que le tri croissant soit appliqué.



Cliquez sur le bouton OK pour valider.
Désormais, l'aperçu du publipostage affiche les lettres par ordre croissant du champ "Montant".

6. Description de la barre d'outils "Fusion et Publipostage"

1. Préparation du document principal:
Ce bouton permet de choisir le type de document pour le publipostage (Lettre, Message électronique, Enveloppe ...)

2. Ouvrir la source de données: permet de choisir une base de données.

3. Fusion et publipostage: Destinataires.
Permet d'afficher la boîte de dialogue pour trier et filtrer les destinataires.

4. Insérer un bloc d'adresse.

5. Insérer une ligne de salutations.

6. Insérer les champs de fusion:
La boîte de dialogue liste tous les champs de la base de données. Sélectionnez un nom dans la liste et cliquez sur le bouton "Insérez". Le champ est positionné à l'emplacement du curseur.

7. Insérer un mot clé:
Les options du bouton sont décrites en détail dans le chapitre VII.

8. Mode publipostage:
Cliquez sur ce bouton pour visualiser le nom du champ dans la lettre type. Recliquez sur le bouton pour afficher la valeur de l'enregistrement.

9. Mettre les champs de fusion en surbrillance.

10. Faire correspondre les champs.

11. Propager les étiquettes.

12. Faire défiler les enregistrements dans le document principal.

13. Rechercher une entrée:
Cette option permet de rechercher un enregistrement à partir d'un mot clé. Il est possible de filtrer la recherche sur un champ spécifique.
Si une entrée est trouvée, l'enregistrement s'affiche dans le document principal.
Vous pouvez ensuite recliquer sur le bouton"Suivant" afin de contrôler s'il existe un autre enregistrement répondant à la requète.

14. Vérifier la fusion:
Cette option permet de simuler ou d'effectuer la fusion et de récupérer un compte rendu des erreurs.

15. Fusionner vers un autre document.

16. Fusionner vers l'imprimante.
Cette option permet d'imprimer tous les enregistrements, l'enregistrement actif ou une sélection filtrée par numéro d'index.

17. Fusionner avec un message electronique: Consultez l'étape 6 du chapitre VIII.

18. Fusionner avec une télécopie:
Consultez le chapitre XII (Les ressources Microsoft) pour plus de détails.

7. Description du bouton "insérer un mot clé"

L'option Demander:
Permet de paramétrer une boîte de dialogue qui va s'afficher au moment de la fusion. Vous pourrez ainsi ajouter une information complémentaire dans chaque lettre, à l'emplacement d'un signet que vous aurez préalablement créé.
"Invite" correspond à la description de la boite de dialogue.
Le texte par défaut est facultatif.

Si vous ne cochez pas l'option "Demander une seule fois", la boîte de dialogue devra être validée autant de fois qu'il y a d'enregistrements dans la fusion.

L'option Remplir:
Le principe est identique à l'option "Demander", mais l'information est ajoutée à l'emplacement du champ "Remplir" (FILLIN).

L'option Si ...Alors ...Sinon...:
Permet d'insérer un texte conditionnel en fonction de la donnée contenue dans un autre champ.

Par exemple, si vous basculez en mode de champ, vous obtenez:

{IF 18 < 0 "A créditer" "A facturer"}

18 est la valeur de l'enregistrement actif.

D'autres informations sur le site Microsoft pour personnaliser les champs conditionnels : Lien35

L'option Numéro enregistrement de fusion:
Insère un champ pour afficher le numéro d'enregistrment dans le document (Equivalent du champ MERGEREC).
Le numéro d'enregistrement correspond à l'ordre dans la base de données. Le numéro peut donc ne pas être chronologique si vous avez trié ou filtré les enregistrements dans la fusion.

L'option Numéro séquence de fusion:
Contrairement à l'option précédente, les numéros sont attribués par ordre chronologique de fusion.

L'option Suivant:
Permet de fusionner l'enregistrement suivant sur la meme feuille (Par défaut il y a toujours un saut de page entre chaque enregistrement).

L'option Suivant Si:
Il s'agit de la même chose que l'option précédente, mais de façon conditionnelle.

Si vous basculez ensuite en mode de champ vous obtenez :{NEXTIF 1 = 5}
1 est la valeur de l'enregistrement actif.
= 5 est la condition définie.

L'option Définir Signet:
Permet d'insérer un signet à l'emplacement du curseur.

L'option Sauter l'enregistrement Si:
Permet d'enlever des enregistrements en appliquant un filtre.
Par exemple, ne pas afficher les enregistrements si le champ "leNom" est égal à "mimi".

9. Utiliser la liste des contacts Outlook pour le publipostage

A l'étape 3 de l'assistant, choisissez l'option "Sélection à partir des contacts Outlook".
Ensuite cliquez sur le bouton "Choisir le dossier contacts".
Sélectionnez le dossier des contacts dans la boite de dialogue, puis cliquez sur le bouton OK pour valider.

Pour le reste, la méthode de préparation du document principal reste similaire aux exemples précédents. Vous pouvez utiliser la liste des contatcts outlook pour envoyer des messages électroniques mais aussi pour créer des lettres, des enveloppes ou des étiquettes.

Si votre objectif est d'envoyer des mails:
Lorsque vous aurez cliqué sur le bouton "Message électronique" à l'étape 6, Sélectionnez le champ "Adresse_électronique" Dans la zone "A". Ce champ correspond aux Adresses de messagerie saisies dans la base des contacts Oulook.

10. Automatiser les tâches de publipostage

Il est possible d'automatiser le publipostage en pilotant la fusion par programmation (VB et VBA). Lorsque la mise en page du document est figée, et si la tâche de publipostage doit être régulièrement exécutée, vous pouvez créer une procédure que se chargera de lancer les impressions.

Cet exemple ouvre le document principal de publipostage Word depuis Excel. La procédure lance ensuite l'impression pour l'ensemble des enregistrements.

Vba
```
Private Sub commandButton1_Click()
    'Nécessite d'activer la référence "Microsoft Word
xx.x Object Library"
    Dim docWord As Word.Document
    Dim appWord As Word.Application

    Application.ScreenUpdating = False
    Set appWord = New Word.Application
    appWord.Visible = False
    'Ouverture du document principal Word
    Set docWord =
appWord.Documents.Open("C:\leDocument.doc")

    'fonctionnalité de publipostage pour le document
spécifié
    With docWord.mailMerge
        'Spécifie la fusion vers l'imprimante
        .Destination = wdSendToPrinter
        .suppressBlankLines = True
            'Prend en compte l'ensemble des
enregistrements
            With .DataSource
                .firstRecord = wdDefaultFirstRecord
                .lastRecord = wdDefaultLastRecord
            End With
        'Exécute l'opération de publipostage
        .Execute Pause:=False
    End With

    Application.ScreenUpdating = True

    'Fermeture du document Word
    docWord.Close False
    appWord.Quit
End Sub
```

Vous pouvez aussi appliquer des filtres lors d'un publipostage automatisé.
Cette macro Excel imprime uniquement les enregistrements dont le champ "Montant" est supérieur à 50.

Vba
```
Private Sub commandButton1_Click()
    'Nécessite d'activer la référence "Microsoft Word
xx.x Object Library"
    Dim docWord As Word.Document
    Dim appWord As Word.Application

    Application.ScreenUpdating = False

    Set appWord = New Word.Application
    appWord.Visible = False

    'Ouverture du document principal Word
    Set docWord =
appWord.Documents.Open("C:\leDocument.doc")

    'Création de la requête:
       'N'oubliez pas d'ajouter le symbole $ après le
nom de la feuille
    docWord.MailMerge.DataSource.QueryString = _
                    "SELECT * FROM [Feuil1$] WHERE
[Montant] > 50"

    'Fusion
    With docWord.MailMerge
        .Destination = wdSendToPrinter
            With .DataSource
                .FirstRecord = wdDefaultFirstRecord
                .LastRecord = wdDefaultLastRecord
            End With
        .Execute Pause:=True
    End With

    'Fermeture du document Word
    docWord.Close False
    appWord.Quit

    Application.ScreenUpdating = True
End Sub
```

Consultez aussi la source de Xo pour créer une liste de publipostage, à partir de Word et d'un fichier texte : Lien36

Retrouvez l'article complet de SilkyRoad en ligne : Lien37

Vu dans la FAQ VBA

A quoi sert l'événement Calculate de la feuille ?

Je serais tenté de répondre "à rien". Celui-ci intervient après le recalcul, ce qui fait qu'il a peu d'utilisations concrètes. Néanmoins il permet de déclencher une opération nécessitant d'attendre la fin du recalcul

Comment insérer un signet qui ne contient pas de texte ?

On appelle la méthode collapse de la sélection avant l'insertion

```
selection.Collapse Direction:=wdCollapseEnd
With ActiveDocument.Bookmarks
    .Add Range:=Selection.Range, Name:="Nom"
    .DefaultSorting = wdSortByName
    .ShowHidden = True
End With
```

Comment scinder un tableau qui fait plus de 10 lignes en dupliquant les titres ?

La manipulation est un peu particulière. Si on travaille uniquement avec les collections exposées par le tableau, on n'accède pas à des méthodes comme Copy ou Paste. Par contre, l'objet Selection expose ces méthodes.

```
Set objTable = ThisDocument.Tables(1)
If objTable.Rows.Count > 10 Then
    objTable.Rows(1).Select
    Selection.Copy
    objTable.Rows(11).Select
    Selection.Paste
    objTable.Rows(11).Select
    Selection.SplitTable
End If
```

Delphi

Les derniers tutoriels et articles

Les interfaces d'objet sous Delphi

Vous trouverez dans ce tutoriel les bases nécessaires à la compréhension du fonctionnement et de la manipulation des interfaces d'objet sous Delphi, communément appelé Interface.

1. Rappels

1.1. Méthode abstraite

Une méthode abstraite est une méthode virtuelle ou dynamique n'ayant pas d'implémentation dans la classe où elle est déclarée. Son implémentation est déléguée à une classe dérivée.

1.2. Classe abstraite

Une classe abstraite est une classe n'ayant pas d'implémentation et ne pouvant être instanciée. Son implémentation est déléguée à une classe dérivée. Avant Delphi 2006 une classe abstraite contenait uniquement des méthodes abstraites, depuis Delphi 2006 le mot clé Abstract peut être associé à une définition de classe

2. Qu'est-ce qu'une interface ?

Ici le mot interface n'a aucun rapport ni avec une interface utilisateur (IHM) ni avec la section interface d'une unité Delphi.
Une interface est une abstraction. Elle est une spécification formelle de classe et est utilisée pour définir la limite entre sa spécification et son implémentation, on laisse ainsi aux classes la liberté de son implémentation.
En d'autre termes une interface est le quoi et une classe définit le comment.

Au niveau du langage Delphi, une interface est un type semblable à une classe abstraite, elle ne contient que des méthodes abstraites par défaut, il n'est pas nécessaire de spécifier le mot clé abstract dans leurs déclarations.
Une interface ne décrit aucune structure de donnée car elle ne peut contenir aucun attribut, elle n'a aucune donnée.
En revanche elle peut contenir des propriétés aux travers d'appel de méthodes d'accesseurs (Set et Get).
Elle est une possibilité d'encapsulation complète.

Les interfaces Delphi ressemblent fortement à celles de Java. Il y a cependant quelques différences au niveau de l'implémentation, en particulier la forte orientation COM des interfaces.

3. A quoi servent-elles ?

Une interface défini une aptitude à faire quelques chose, par exemple une itération sur un ensemble de données, ou un comportement de déplacement dans un espace 2D.
Un des objectifs des interfaces est de diminuer le couplage entre deux classes, c'est à dire le degré de dépendance d'une classe par rapport à une autre, ce qui facilite la maintenance.

L'interface permet de ne présenter au client que ce qui l'intéresse, dans ce tutoriel nous l'illustrerons en abordant les méthodes pouvant être liées au déplacement d'un être ou d'une chose.

L'usage d'une interface à la place d'une classe, permet de modifier l'implémentation de l'abstraction, ici le déplacement, sans impacter la ou les classes utilisatrices, l'interface faisant office d'écran.

Prenons un exemple, généralement la notion de déplacement, commune aux êtres ou aux objets, tombe sous le sens en revanche sa spécialisation au sein de la classe serpent ou flèche d'arc est radicalement différente. Une interface permettrait ici, dans un contexte précis, de s'intéresser à l'essentiel : ce que ça fait et pas comment ça le fait.

En quelque sorte une interface répond à la question que l'on pourrait poser à une classe : Est-ce que tu sais faire ça ?
La réponse qui nous importe ici est oui ou non, et c'est tout. La manière de le faire, c'est à dire comment est rempli le contrat, nous importe relativement peu.

Le terme de contrat est souvent employé pour définir la convention par laquelle une interface interagit avec une classe mais ce qui importe c'est plus le respect de ce que l'interface est sensée faire que l'acte qui enregistre cette convention. Dans la vie on peut prendre des engagements ou signer tous les contrats que l'on veut, rien ne nous empêche de ne pas les respecter. Je serais tenté de dire plus simplement qu'une interface respecte toujours l'engagement pris sur le comportement qu'elle propose et dans le cas contraire la sanction est simple et sans appel, la compilation échoue.

4. Comment ça marche ?

Chargez le projet interface1.

Par convention, le nom d'une interface est préfixé par la lettre I en majuscule et son nom défini clairement son comportement.
Déclarons une interface proposant la capacité de se déplacer :

```
Type
IDeplacable =Interface
   Procedure Deplace; // Méthodes abstraites par défaut
   Procedure Arret;
end ;
```

On utilisera, par exemple la méthode Deplace non pas via une instance de classe mais au travers d'une variable du type de l'interface souhaitée, on parlera donc dans ce cas de référence d'interface.

Sachez qu'une variable d'interface, ici la variable MonInterface, est initialisée à Nil lors de sa déclaration :

```
Procedure TestInterface(UneInterface : IDeplacable);
begin
 If assigned(UneInterface)
  then Writeln('L''interface est assignee.')
  else Writeln('L''interface n''est pas assignee.');
end;

var MonInterface : IDeplacable;
begin
 TestInterface(MonInterface);
end.
```

Créons une interface en utilisant le code suivant :

```
var MonInterface : IDeplacable;
begin
 MonInterface:=IDeplacable.Create;
end.
```

malheureusement ce code provoque l'erreur de compilation : "Type record, object ou class requis".

Attention : La réalisation d'une référence d'interface ne peut se faire en dehors de celle d'une instance de classe. Pour agir sur une interface on doit impérativement créer une instance d'une classe implémentant le type d'interface attendue.
Les classes sont donc responsables de l'implémentation de la ou des interfaces qu'elles supportent.

Afin d'utiliser notre interface IDeplacable nous devons déclarer une classe l'implémentant :

```
  TChose= Class(TObject, IDeplacable)
   Distance : Integer
  end;
```

Le premier type déclaré doit être impérativement une classe, ici TObject, suivie d'une ou plusieurs interfaces, ici IDeplacable.
Ajoutons la création de l'instance de la classe TChose :

```
begin
 TestInterface(MonInterface);
 UneChose:=TChose.Create;
 readln;
end;
```

La compilation du code précédent nous renvoie, entre autres, les erreurs suivantes :

```
Identificateur non déclaré : Deplace
Identificateur non déclaré : Arret
```

Elles confirment que les méthodes de l'interface IDeplacable sont bien abstraites et doivent être implémentées par la classe TChose. Les simples déclarations des méthodes de l'interface sont insuffisantes, dans ce cas on retrouve une erreur classique : "Déclaration forward ou external non satisfaite : ' TChose.Deplace'".

Implémentons donc les deux méthodes de l'interface :

```
 TChose= Class(TObject,IDeplacable)
   Distance : Integer;
      //IDeplacable
   Procedure Deplace(Const Param:String);
   Procedure  Arret;
      //TChose
End;

{ TChose }
procedure TChose.Arret;
begin
 Writeln('Arret : IDeplacable de TChose ');
end;

procedure TChose.Deplace(const Param: String);
begin
  Writeln('Deplace : IDeplacable de TChose ');
end;
```

Ici la compilation nous renvoie la même erreur mais pour les méthodes nommées QueryInterface, _AddRef et _Release. Bien que nous respections le contrat de l'interface IDeplacable, notre classe est soumise au respect du contrat de l'interface héritée IInterface. Du fait que, comme une classe, une interface hérite de toutes les méthodes de ces ancêtres on doit implémenter toutes ces méthodes.

Remarque : Il reste possible de ne pas implémenter totalement une méthode d'interface, c'est à dire que le corps de la méthode peut ne rien faire.

5. L'interface IInterface

IInterface est l'ancêtre de base de toutes les interfaces, les déclarations suivantes :

```
type
 IDeplacable =interface

 IDeplacable =interface(IInterface)
```

sont donc identiques.

Voici la déclaration de l'interface IInterface :

```
type
  IInterface = interface
    ['{00000000-0000-0000-C000-000000000046}']
    function QueryInterface(const IID: TGUID; out Obj):
HResult; stdcall;
    function _AddRef: Integer; stdcall;
    function _Release: Integer; stdcall;
  end;
```

Elle permet de gérer ou créer des interfaces COM sous Delphi. On peut également voir dans le code source de l'unité System.pas la déclaration suivante

```
 IUnknown = IInterface;
```

Les trois méthodes de IInterface sont donc celles de IUnknown. Sous Delphi hériter de IUnknown au lieu de IInterface informe le compilateur que l'interface doit être compatible avec les objets COM.

6. L'interface inconnue (IUnknown)

Chargez le projet Interface2.
L'interface IUnknown est l'ancêtre des interfaces sous COM et possède son propre identifiant d'interface(IID), chaque objet COM doit l'implémenter. Les méthodes de l'interface IUnknown permettent de gérer le cycle de vie des objets et d'obtenir les autres interfaces gérées par le composant.

Dans la déclaration de IInterface, la partie optionnelle ['{00000000-0000-0000-C000-000000000046}'] spécifie un GUID qui permet d'identifier de manière unique une interface. Ce GUID est nécessaire pour l'interrogation d'interface via l'appel à la méthode QueryInterface.

6.1. Un compteur de références

Les interfaces COM utilisent un compteur de références pour chaque instance d'objet afin de contrôler sa durée de vie. Une des raisons est que plusieurs clients peuvent utiliser une même référence d'interface.

Lorsqu'une référence d'interface est demandée, par exemple lors d'une assignation, le compilateur insère un appel à la méthode _AddRef qui incrémente de 1 son compteur de référence. Une fois que la référence d'interface n'est plus utilisée, par exemple lors de l'affectation à NIL, le compilateur insère un appel à la méthode _Release qui décrémente le compteur de référence, et si celui-ci est nul, le destructeur de l'instance d'objet implémentant l'interface est appelé.

On peut donc avoir plusieurs références d'une même interface pour une seule instance d'objet. Sans ce mécanisme il serait impossible de savoir quand détruire l'objet implémentant la ou les interfaces manipulées.

Attention : L'usage du compteur de références comporte quelques pièges comme nous le verrons par la suite.

6.2. La méthode QueryInterface

Elle permet d'obtenir un pointeur d'interface pour l'interface identifiée par le paramètre IID. Si l'objet supporte l'interface requise, il est renvoyé dans le paramètre Obj et la méthode QueryInterface renvoie S_OK. Si l'objet ne supporte pas l'interface, QueryInterface renvoie E_NOINTERFACE.

6.3. La méthode _AddRef

Incrémente le compteur de références de cette interface.
Une interface peut proposer ses services à plusieurs clients, chaque client devra donc posséder une référence sur cette interface.

6.4. La méthode _Release

Décrémente le compteur de références de cette interface. Chaque client doit donc libérer la référence de chaque interface utilisée. Une fois le compteur de référence à zéro l'interface libère l'instance de classe sous-jacente.

Remarque : Sous Delphi .NET ces méthodes sont facultatives sauf si votre interface doit être accessible via COM.

7. Détails d'implémentation d'une interface

Chargez le projet Interface3.
Revenons à notre code d'origine et utilisons la procédure TestInterface :

```
Procedure TestInterface(UneInterface : IDeplacable);
begin
 If assigned(UneInterface)
  then
   begin
    Writeln('L''interface est assignee.');
    UneInterface.Deplace('Méthode d''''interface''');
   end
  else Writeln('L''interface n''est pas assignee.');
end;
```

Appelons la avec en paramètre une instance de classe au lieu d'une référence d'interface.

```
UneChose:=TChose.Create;
TestInterface(UneChose);
readln;
```

On peut voir qu'il n'y a aucun problème particulier ni lors de la compilation ni lors de l'exécution. Par contre l'ajout, dans la procédure TestInterface, des lignes suivantes :

```
UneInterface.ProcedureDeTChose;
Inc(UneInterface.Distance,10);
```

provoque, lors de la compilation, l'erreur : "Identificateur non déclaré".

L'interface ne connaît rien de l'objet associé car comme il a été dit précédemment on s'attache à l'essentiel en respectant le contrat passé. En revanche l'implémentation d'une méthode d'une interface peut manipuler des données de l'instance de classe :
procedure Tchose.Deplace;

```
begin
  Writeln('Deplace : IDeplacable de TChose');
  Inc(Distance,10);
end;
```

On peut donc dire que TChose.Deplace est égale à IDeplace.Deplace mais pas que TChose est égale à IDeplace.

On ne peut pas transtyper IDeplace en TChose en revanche TChose peut être transtypé en IDeplace, plus précisément il s'agit d'une conversion de type implicite. C'est ce qui se passe lors de l'appel à la procédure TestInterface, bien que cela soit le compilateur qui effectue cette opération à notre place. Il ne s'agit pas vraiment d'un transtypage car le type de l'interface est connu lors de la compilation. Il n'y a donc pas d'interrogation de la table des méthodes des interfaces (IMT).

Le code suivant crée un objet TChose puis récupère son interface IDeplacable qui permet l'appel de la méthode Deplace via la variable MonInterface et enfin décrémente le compteur de référence de l'interface l'interface.

```
UneChose:=TChose.Create;
MonInterface:=UneChose;  //Affectation
                         //transtypage implicite lors
de la compilation
MonInterface.Deplace;
MonInterface:=Nil;
UneChose.Free;
```

Attention : Ici l'assignation de NIL dans la variable d'interface ne provoque aucune suppression de l'instance de classe car, bien qu'il soit en théorie à zéro, le compteur de référence n'est pas géré. Notez toutefois que l'appel des méthodes _AddRef et _Release est effectif.

```
Appel de TChose._AddRef
Deplace : IDeplacable de TChose
Appel de TChose._Release
```

7.1. Les problèmatiques de transtypage des interfaces

Chargez le projet40.
Essayons de manipuler une interface IDeplacable sur une instance de la classe TObject :

```
var MonInterface : IDeplacable;
    UnObjet :TObject;

begin
 UnObjet:=TObject.Create;

{1}  MonInterface:=UnObjet;
{2}  if UnObjet is IDeplacable then writeln('Un objet
est une interface IDeplacable.');
{3}  MonInterface:=UnObjet as IDeplacable;
{4}  MonInterface:=IInterface(UnObjet) as IDeplacable;
{5}  MonInterface:=IUnknown(UnObjet) as IDeplacable;
{6} if Not Supports(UnObjet,IDeplacable)
     then  Writeln('la classe TObject ne supporte pas
l''interface IDeplacable.');

 UnObjet.Free;
 readln;
end.
```

La ligne 1 provoque l'erreur de compilation : "Type incompatible IDeplacable et TObject."
TObject ne supporte pas l'interface IDeplacable, le transtypage implicite ne peut se faire.

La ligne 2 provoque l'erreur de compilation : "Opérateur non applicable à ce type d'opérande."
L'opérateur IS est dédié aux classes uniquement.

La ligne 3 provoque l'erreur de compilation : "Opérateur non applicable à ce type d'opérande."
L'interface IDeplacable ne possède pas de GUID, le transtypage ne peut donc se faire.

La ligne 4 et 5 provoque l'erreur de compilation : "Type incompatible IInterface et TObject."
TObject ne supporte pas l'interface IDeplacable, le transtypage explicite à l'aide de l'opérateur AS ne peut se faire.

Ces erreurs de compilation mettent en évidence qu'une classe ne peut proposer que les interfaces qu'elle implémente, TObject n'en propose aucune.

Il est possible de savoir si une classe supporte ou pas une interface donnée, Delphi propose à cette fin la fonction Supports :

```
 if Not Supports(UnObjet,IDeplacable)
  then  Writeln('la classe TObject ne supporte pas
l''interface IDeplacable.');
```

Ce code provoque l'erreur de compilation : L'interface 'IDeplacable' n'a pas d'identification d'interface.
Ajoutons un GUID (un identifiant unique d'interface), par la combinaison de touche Shift-Ctrl-G :

```
  IDeplacable=interface(IInterface)
  ['{62CAE27F-94C1-4A3D-B94F-F08FF36207D5}'] // GUID
nécessaire pour l'opération de cast
```

A partir de là l'appel de la fonction Supports réussit. En revanche les lignes de code 1 à 5 restent erronées (nous reviendrons sur l'erreur de la ligne 3).

Remarque : Sous Delphi .NET l'opérateur is est valide sur une interface et l'usage de l'opérateur as ne nécessite pas de GUID, sauf si votre interface doit être accessible via COM.

Chargez le projet Interface41.
Reprenons notre classe TChose pour opérer ce transtypage d'interface et ajoutons une seconde déclaration d'interface :

```
type
  IMesurable=interface(IInterface)
   Function Dimension:Integer;
  End;

...
var MonInterface : IDeplacable;
    Taille : IMesurable;
    UneChose: TChose;
begin
 UneChose:=TChose.Create;
 if Not Supports(UneChose,IDeplacable)
  then  Writeln('la classe TChose ne supporte pas
l''interface IDeplacable.');

 MonInterface:=UneChose as IDeplacable;
```

Bien que l'interface IDeplacable possède un GUID, la dernière ligne provoque l'erreur de compilation : "Opérateur non applicable à ce type d'opérande."
Ici on appelle une des fonctions Supports surchargée avec une instance de classe, cette méthode appelle en interne la méthode héritée TObject.GetInterface le traitement peut donc se faire sans problème puisqu'elle parcourt la liste des interfaces de l'instance UneChose.
Par contre l'opérateur AS implique que la classe concernée implémente la méthode QueryInterface afin d'obtenir un pointeur d'interface pour l'interface identifiée par le paramètre IID (c'est à dire IDeplacable).
Ce qui est bien le cas dans notre exemple! Mais que se passe-t-il donc ?

Essayons de modifier la déclaration de la classe TChose ainsi :

```
  //TChose= Class(TObject,IDeplacable)
  TChose= Class(TInterfacedObject,IDeplacable)
```

La compilation réussit.
La raison en est que la classe TInterfacedObject implémente l'interface IUnknown, avec cette déclaration le compilateur sait que la classe TChose implémente bien la méthode QueryInterface.
Dans notre déclaration d'origine IDeplacable hérite de IInterface mais masque au compilateur cette information.

Déclarons explicitement cette interface dans la liste d'interfaces de notre classe:

```
  TChose= Class(TObject,IInterface,IDeplacable)
```

Ici aussi la compilation réussit.
On peut donc ainsi créer une interface identifiée, supportant le transtypage et sans gestion du compteur de références ce qui n'est pas le cas si on utilise la classe de base TInterfacedObject au lieu de TObject.

Attention l'usage de l'opérateur As sur une interface incrémente le compteur de référence :

```
 Appel de TChose.QueryInterface
 Appel de TChose._AddRef
```

Pour terminer abordons la gestion de la seconde interface qui dispose d'un GUID :

```
Taille:=UneChose as IMesurable;
```

Ce transtypage provoque à l'exécution l'exception EIntfCastError. On doit donc utiliser un bloc Try..Except pour la gérer

```
try
  Taille:=UneChose as IMesurable; // cf. la procédure
interne à Delphi System._IntfCast
  except
  On E :EIntfCastError do
   begin
    Writeln('Exception :');
    Writeln(E.Message);
   end;
end;
```

Attention : Vous remarquerez qu'ici le compteur de références n'est pas appelé.

```
Appel de TChose.QueryInterface
Exception :
Interface non supportée
```

Retrouvez la suite de l'article de Laurent Dardenne en ligne : Lien39

Entretien avec Florent Ouchet

Entretien avec Florent Ouchet le 2 décembre 2006.

La « Jedi Code Library » est un recueil de fonctions et de classes pouvant être réutilisées dans beaucoup d'applications.
La JCL est une bibliothèque du Projet JEDI. Redistribuée sous la « Mozilla Public Licence » (MPL), son code peut être utilisé aussi bien dans les applications commerciales (avec ou sans disponibilité des sources) que dans d'autres projets «open source».

Il répond aux questions à propos du mode de fonctionnement de l'équipe en charge de la librairie JCL, questions proposées par les membres de l'équipe Delphi de developpez.com.

Equipe Delphi de developpez.com: Quelles sont vos responsabilités au sein du projet JCL ?

Florent Ouchet: Une traduction du rôle serait "Coordinateur de l'équipe JCL", il s'agit de donner au projet les directions principales de son développement. Je m'occupe aussi des tâches administratives du projet : création et publication des fichiers, annonces sur les forums. J'exerce cette fonction depuis le début de l'année 2006 en prenant la succession de Robert Rossmair. En dehors des tâches administratives, je participe activement au développement de la librairie.

Combien de personnes travaillent sur ce projet ?

Ce nombre est assez difficile à définir. Dans un projet «open source», tout développeur peut travailler sur le projet en partant du code existant. De nombreuses modifications sont ensuite données par leurs auteurs afin d'être intégrées dans le code du projet.
L'équipe des membres de la JCL comporte actuellement 15 développeurs de nationalités différentes qui peuvent valider les changements dans le code. Elle peut être consultée ici. Cette équipe n'est pas figée et l'ajout de nouveaux membres arrive fréquemment.

Quelles sont les difficultés que vous rencontrez dans la gestion de ce projet ?

La principale limitation est le temps que je peux consacrer à ce projet, il faut le partager entre l'aide aux utilisateurs, la maintenance du code, le suivi des bogues et l'implémentation de nouvelles fonctionnalités. Le projet compte actuellement environ 8Mo de code (peut-être 100 000 lignes de code), le faible nombre de membres limite le développement actuel. Un autre problème est la barrière de la langue : la quasi-totalité des communications sont en anglais, le principal est d'arriver à se faire comprendre.

Comment devient-on contributeur de la JCL ?

Il faut bien sûr être un utilisateur de la JCL et se montrer volontaire. On peut séparer les contributeurs ponctuels qui nous soumettent des modifications ou des ajouts de code et les utilisateurs qui viennent avec des idées constructives et un projet. Dans le premier cas, un membre se charge de l'intégration du code en respectant les droits du contributeur comme décrits dans la MPL : son nom est ajouté à la liste des contributeurs du fichier, le contributeur reste l'auteur de ses modifications. Le deuxième cas est arrivé très récemment quand des utilisateurs nous ont fait part de propositions constructives et de projets concernant les PCRE (Expression Régulière Compatible « Perl »), je leur ai proposé de devenir membres.

Quelle est la motivation principale des contributeurs ?

La motivation est principalement le développement et le suivi de la bibliothèque. Tous sont des passionnés de programmation et de Delphi, chacun trouve un intérêt à contribuer au développement. Certains développeurs sont des professionnels qui consacrent une partie de leur temps de travail à améliorer le code de la bibliothèque qu'ils utilisent dans leurs projets.

Avez-vous une démarche particulière concernant l'intégration dans l'équipe ?

Les membres du projet doivent être actifs et présents régulièrement sur le forum de discussion.

Existe-t-il par exemple un pré-requis minimum (technique ou autre) pour vous rejoindre ?

Il n'y a pas de règle stricte, les développeurs sont jugés sur leur motivation et la qualité de leur code.

Existe-t-il un turn-over important dans l'équipe ? Si oui, pose-t-il des problèmes particuliers ?

Certains membres sont présents depuis la création, leur expérience est très instructive. Je n'ai pas eu jusqu'à présent de problème qui n'aurait pu être résolu que par un ancien développeur.

Comment se prennent les décisions importantes ?

Les décisions importantes sont prises en concertation, une question est posée sur le forum de discussion et tous les

développeurs sont invités à donner et à discuter leur avis, un compromis arrive toujours à être accepté.

Comment procédez-vous pour le choix des composants à intégrer ?

Ils sont choisis selon leur utilité pour les utilisateurs de la bibliothèque et leur compatibilité avec les outils de développement et les environnements d'exécution. Un composant spécifique à une seule version de Delphi ou une seule version de Windows ne sera pas intégré. Avant leur intégration dans la bibliothèque, leur code est relu et corrigé afin de satisfaire les règles d'écriture de code et de les intégrer de manière optimale (élimination des redondances avec du code existant, regroupement des constantes...)

Quels sont vos liens avec Borland ?

Nous sommes des utilisateurs de produits Borland. Suite à la création de CodeGear, nous supporterons les nouveaux EDI créés par cette nouvelle entité. Depuis Delphi 2005, du code de la JCL est utilisé dans l'EDI de Delphi/BDS pour récupérer les piles d'appel et les informations de débogages quand une exception provoque un « crash » récupérable.

Si l'organisation d'un projet open source se différencie d'un projet entreprise, en quoi diffère-t-elle ?

Les moyens mis en œuvre ne sont pas comparables ; le coté commercial apporte des moyens organisationnels, financiers et humains sans commune mesure avec le monde open-source où la plupart des contributions sont faites sur la base du volontariat. Un projet commercial peut plus facilement s'engager sur des résultats en un temps donné, une équipe de projet open-source agit selon les besoins exprimés par les utilisateurs et les envies de chacun de ses développeurs. Un projet d'entreprise ne laisse pas autant de liberté individuelle car les développeurs doivent absolument respecter les besoins et les délais décidés avec le client. Néanmoins la frontière entre les projets libres et les projets d'entreprise tend à disparaître car certaines grandes firmes font d'un projet open-source leur projet en investissement massivement dans son développement.

Comment les membres de l'équipe communiquent-ils ?

Tous les outils classiques d'Internet sont utilisés (mail, messageries instantanées, forums de discussions).

Utilisez-vous un outil de travail collaboratif ?

Nous utilisons principalement Subversion pour gérer les versions des sources. Cet outil permet à un membre du projet de valider des changements pour qu'ils puissent instantanément être utilisés par tous les autres développeurs autour du monde. Cet outil propose les fonctions de comparaison entre les versions, il est aussi possible d'avoir tous les détails sur les changements effectués depuis les trois dernières années. Le suivi des bogues est facilité par Mantis qui permet d'organiser et de discuter les solutions pour les résoudre.

Quels sont vos projets ?

Il pourrait être intéressant de regarder du coté de FreePascal/Lazarus pour permettre à cette communauté d'utiliser la JCL.

Nous avons pu voir dans les derniers builds quotidiens que la prochaine version de la JCL envisage d'intégrer un expert pour Subversion, est-ce bien le cas ?

Cet expert (nom pour une extension de l'EDI) vise à faciliter les connexions avec un serveur de sources CVS ou Subversion. Il ajoute à l'interface des EDI de Borland un nouveau menu et des fonctions pouvant être placées dans les barres d'outils. Toutes les opérations courantes (comme la mise à jour, les validations…) peuvent être réalisées directement depuis l'EDI. Son fonctionnement requiert l'installation sur la machine de TortoiseSVN ou de TortoiseCVS car il utilise ces programmes pour exécuter les actions. Il peut déjà être testé et utilisé en utilisant les versions de développement de la JCL : les zips quotidiens qui peuvent être téléchargés à l'adresse http://jcl.sourceforge.net/daily/ ou en se connectant au serveur de source hébergé par Sourceforge.

Delphi .NET ne semble pas avoir le même succès que Delphi Win32, à quoi est-ce dû selon-vous ?

Les clients de Borland/CodeGear apprécient la régularité des produits, le fait de changer de cible tous les 3 ou 4 ans fait peur ; la majorité des développeurs Delphi a décidé de ne pas suivre la mode, qui va encore changer dans quelques années, remettant en question une fois encore la compatibilité du code. Les processeurs seront toujours capables d'exécuter du code natif (le code managé est une surcouche artificielle) et Microsoft devra pour ne pas se couper de la majorité de ses utilisateurs conserver et améliorer le support natif. Le Delphi Win32 et bientôt le Delphi 64 bit ont encore un long avenir.

Au vue du nombre de développeurs, quelles règles particulières avez-vous mises en place concernant la gestion des sources sous CVS ?

Depuis le mois d'avril, les sources de la JCL sont gérées par Subversion. La clarté du code est un point clé car il doit pouvoir être lu et compris par tous les développeurs désirant comprendre le fonctionnement.

Les règles étaient déjà en place auparavant :

- écriture de code robuste
- succès de la compilation
- respect de la syntaxe standard (comme dans la RTL et la VCL)
- la langue anglaise dans les sources
- tester le fonctionnement du code

Comment se fait-il que les composants et le code source soient si peu documentés ?

C'est un problème récurrent dans beaucoup de projets «open source», les développeurs se focalisent sur l'écriture de code et oublient la documentation pour les utilisateurs. Il est très dur pour un développeur d'écrire la documentation pour son code car la vision interne qu'il a de son code est très différente des attentes d'un utilisateur recherchant des informations sur la boite noire. Le développement du projet est basé sur le volontariat, tous les développeurs peuvent écrire la documentation ou les commentaires s'ils le souhaitent.

Comment se passe la maintenance dans ce cas ?

La dénomination des différentes fonctions, variables ou classes est très importante dans la compréhension d'un code existant. La maintenance du code passe une étape préliminaire de prise en main du code existant qui serait nécessaire même si le code était

commenté. La manière de commenter et de lire les commentaires est très spécifique à chaque programmeur, les commentaires laissés par le développeur précédent seraient d'une faible utilité dans la compréhension du code.

Connaissez-vous l'outil DelphiCodeToDoc ? Si oui qu'en pensez-vous ?

Les projets JEDI utilisent Doc-o-matic de Tools Factory. C'est un outil de création de documentation capable de lire les codes sources pour organiser automatiquement les rubriques selon la hiérarchie des classes et la structure du code. Il nous permet de générer très rapidement la documentation sous plusieurs formats (WinHelp, Html, Html Help, MS Help 2.0 ou pourquoi pas PDF).

Quelles sont les dernières évolutions de Delphi qui ont été appréciées par l'équipe JEDI ?

Florent Ouchet:
Les «refactorings» de BDS 2006 facilitent la maintenance de code existant en simplifiant les opérations basiques (extraction de fonctions...). Les vérifications de la syntaxe du code sont bien accélérées par « Error insight » qui évite d'avoir à attendre le temps de compilation. La prochaine version de Delphi promet l'ajout de « templates » de code (un peu comme en C++), cette fonctionnalité pourrait être utilisée pour généraliser certaines portions de code.

La mise à disposition de Turbo Delphi Explorer (gratuit), intégrant un outil de modélisation UML (Together), influencera-t-il le projet JEDI ?

Les nouvelles versions gratuites pourraient créer de nouvelles vocations. Malheureusement dépourvues de compilateur en ligne de commande, nous ne pouvons pas les supporter officiellement. La modélisation formelle de la totalité de JCL est impossible à cause de son code plutôt bas niveau.

Voulez-vous profiter de cet interview pour lancer un appel à candidature ?

Oui, l'équipe de développement de la JCL est ouverte aux nouveaux membres qui souhaiteraient participer à l'effort de développement en améliorant le code existant, en proposant de nouvelles fonctionnalités, en écrivant de nouveaux programmes d'exemple ou en continuant la rédaction de la documentation.

Contact via les news groups : Lien40
(Dans ces news groups utilisez de préférence l'anglais)

Retrouvez l'interview de Florent Ouchet en ligne : Lien41

FAQ Delphi

"Au travers des nombreuses thématiques abordées dans la F.A.Q. Delphi de www.developpez.com, vous trouverez, sans aucun doute, la réponse à votre question qu'elle soit de niveau débutant ou expert. Les réponses de cette FAQ sont tirées des différentes discussions du forum Delphi de www.developpez.com et de propositions ponctuelles de la part de membres.

Ainsi, vous pouvez trouver de précieuses informations sur le fonctionnement et l'utilisation de votre EDI préféré, des trucs et astuces pour des composants de la VCL (bibliothèque de composants visuels de Borland), des aides sur l'utilisation des composants Indy (composants encapsulant les fonctions réseau pour Delphi), etc.
Il serait bien long de dresser ici une liste exhaustive de tous les sujets traités dans les 843 réponses actuellement disponibles."

Comment récupérer le numéro de version de mon application ?

Voici une fonction qui récupère cette information :

```
function ApplicationVersion: String;
var
  VerInfoSize, VerValueSize, Dummy: DWord;
  VerInfo: Pointer;
  VerValue: PVSFixedFileInfo;
begin
  VerInfoSize :=
GetFileVersionInfoSize(PChar(ParamStr(0)), Dummy);
  {Deux solutions : }
  if VerInfoSize <> 0 then
  {- Les info de version sont inclues }
  begin
    {On alloue de la mémoire pour un pointeur sur les
info de version : }
    GetMem(VerInfo, VerInfoSize);
    {On récupère ces informations : }
    GetFileVersionInfo(PChar(ParamStr(0)), 0,
VerInfoSize, VerInfo);
    VerQueryValue(VerInfo, '\', Pointer(VerValue),
VerValueSize);
    {On traite les informations ainsi récupérées : }
    with VerValue^ do
    begin
      Result := IntTostr(dwFileVersionMS shr 16);
      Result := Result + '.' + IntTostr(dwFileVersionMS
and $FFFF);
      Result := Result + '.' + IntTostr(dwFileVersionLS
shr 16);
      Result := Result + '.' + IntTostr(dwFileVersionLS
and $FFFF);
    end;

    {On libère la place précédemment allouée : }
    FreeMem(VerInfo, VerInfoSize);
  end

  else
    {- Les infos de version ne sont pas inclues }
    {On déclenche une exception dans le programme : }
    raise EAccessViolation.Create('Les informations de
version de sont pas inclues');
end;
```

Il faut pour que cela fonctionne que vous spécifiez l'option Inclure les informations de version dans le projet dans menu Projet| Option|Information de version, ainsi la fonction retournera '1.0.0.0' par exemple, sinon la fonction déclenchera une erreur EAccessViolation.

Comment ajouter un évènement à un composant créé dynamiquement ?

Pour ajouter un évènement à un composant créé dynamiquement

il faut procéder manuellement comme Delphi en mode conception.

En premier lieu il faut déclarer l'évènement dans la déclaration de votre classe.

```
type
  TForm1 = class(TForm)
    ...
  public
    { Déclarations publiques }
    { Le nom de la procédure importe peu  }
    { Mais il FAUT respecter les paramètres  }
    Procedure MonClickSurBouton  (Sender: TObject);
  end;


Puis il faut écrire le code de l'évènement :
Procedure TForm1.MonClickSurBouton  (Sender: TObject);
Begin
  ShowMessage('Click !');
End;


Enfin il faut associer l'évènement au moment de la
création des composants :
With TButton.Create(Self) Do
Begin
  Parent := Self;
  Caption := 'Un bouton !';
  { ajout de l'évènement }
  OnClick := MonClickSurBouton;
End;
```

Comment convertir une image JPEG en BMP?

Il suffit pour cela de charger l'image dans un TJPegImage puis de recopier son contenu dans un TBitmap pour enfin l'enregistrer.

```
procedure ConversionJPEGversBMP(const FichierEntree,
FichierSortie: string);
var ImageJPEG : TJPEGImage;
    ImageBitmap : TBitmap;
begin
  ImageJPEG := TJPEGImage.Create;
  try
    ImageJPEG.LoadFromFile(FichierEntree); //Chargement
de l'image

    ImageBitmap := TBitmap.Create;
    try
      //On donne la même taille que l'image jpeg
      ImageBitmap.Width := ImageJPEG.Width;
      ImageBitmap.Height := ImageJPEG.Height;

      //On dessine le jpeg sur le canvas du Bitmap
      ImageBitmap.Canvas.Draw(0, 0, ImageJPEG);
      //On enregistre
      ImageBitmap.SaveToFile(FichierSortie);
    finally
      ImageBitmap.Free;
    end;
  finally
    ImageJPEG.Free;
  end;
end;
```

Comment se connecter à WMI ?

Avant toute chose vous devez importer la librairie de type (fichier .TLB) qui se trouve dans le répertoire du référentiel WMI.
Sous XP :

```
C:\WINDOWS\system32\wbem\wbemdisp.tlb
C:\WINDOWS\system32\wbem\wbemads.tlb (Pour manipuler
Active Directory via WMI)
```

Utiliser pour ce faire le menu "Composant- Importer un contrôle ActiveX".

Une fois ceci fait vous disposerez d'une unité qui vous permettra d'accéder aux objets WMI. Cette unité se trouvera dans le répertoire ..\Delphi\Imports\xxxx_TLB.pas.
Cette unité est à ajouter systématiquement dans la clause uses pour tous les exemples concernant WMI présentés dans cette FAQ.

Pour se connecter à WMI on utilise un composant TSWbemLocator.
Sa méthode ConnectServer établit une connexion au référentiel WMI dans un espace de noms particulier. Dans l'exemple suivant on se connecte dans l'espace de noms ROOT\CIMV2.

La modification du curseur de la souris indique à l'utilisateur la phase de connexion au référentiel WMI.
L'utilisation du flag wbemConnectFlagUseMaxWait évite une attente indéfinie en cas d'indisponibilité de la machine distante.

Si l'appel de la méthode ConnectServer réussit, elle renvoie un objet SWbemServices initialisé, sinon le traitement ne peut se poursuivre.

Cet objet SWbemServices nous permettra d'exécuter des opérations sur l'espace de noms courant.

```
Var
  WMILocator:          TSWbemLocator;
  WmiService:          SWbemServices;

  OldCursor:           TCursor;

begin
 WMILocator:= TSWbemLocator.Create(self);
 try
  OldCursor := Screen.Cursor;
  Screen.Cursor := crSQLWait;

  WmiService:= WMILocator.ConnectServer('.',
'ROOT\CIMV2', '', '', '',
                                        '',
wbemConnectFlagUseMaxWait, nil);
 Finally
  WMILocator.Free;
  Screen.Cursor:= OldCursor;
 end;
end;
```

Si vous déposez un composant TSWbemLocator sur un fiche renseignez respectivement ses propriétés AutoConnect à false et Connectkind à ckRunningOrNew.

Ce code ne contient pas de test d'erreurs. Vous trouverez dans les codes sources proposés avec le second tutoriel sur WMI le projet 1-SWbemLastError\WbemLastError.dpr qui vous permettra de voir dans le détail la gestion des erreurs.

Retrouvez la FAQ Delphi en ligne : Lien42

Les derniers tutoriels et articles

REXML: Parser des documents XML en Ruby

REXML (Ruby Electric XML) est l'outil XML de référence pour les développeurs Ruby, livré en standard avec Ruby. Il est rapide, écrit en Ruby, et peut être utilisé de deux façons : parcours d'arbre ou parcours évenementiel.
Dans cet article, nous allons voir quelques méthodes montrant comment utiliser REXML pour parcourir un XML. Nous allons également voir comment utiliser le debugger interactif de Ruby (irb) pour explorer les documents XML avec REXML. Nous allons utiliser une bibliographie comme exemple de XML.
Vous allez apprendre à parcourir le document via l'API de parcours d'arbre, à accéder aux différents éléments et à leurs attributs, ainsi qu'à créer et insérer des éléments.
Nous allons également voir les particularités des noeuds texte et du traitement des entités. Enfin, nous verrons un exemple d'utilisation de l'API de parcours évenementiel.

1. Débuter avec le parcours d'arbre

Bibliography.xml : Lien43

Nous allons commencer avec l'API de parcours d'arbre, qui ressemble beaucoup au DOM mais en plus intuitif. Voici un premier exemple de code :

code1.rb - Afficher un fichier XML

```
require 'rexml/document'
include REXML
file = File.new("bibliography.xml")
doc = Document.new(file)
puts doc
```

Le **require** charge la librairie REXML. Nous incluons ensuite l'environnement REXML ; nous n'avons plus ainsi à utiliser de noms comme "REXML::Document" tout le temps. Puis nous ouvrons le fichier existant "bibliography.xml" et nous le parcourons en le stockant dans un objet Document. Enfin, nous affichons le document à l'écran.
Quand vous exécutez la commande "ruby code1.rb", le contenu de notre document XML est affiché.

Il est possible que vous obteniez ce message d'erreur :

```
example1.rb:1:in `require': No such file to load
  -- rexml/document (LoadError)
        from example1.rb:1
```

Dans ce cas, c'est dû au fait que REXML n'a pas été installé avec Ruby, ce qui arrive avec certains gestionnaires de packages comme Debian APT qui installent séparément les packages. Installez le package manquant, puis réessayez.

La méthode **Document.new** prend en paramètre des objets de type IO, Document ou String. L'argument spécifie la source à partir de laquelle nous voulons lire le document XML. Dans le premier exemple, nous avons utilisé un objet IO, précisément un objet File qui hérite de la classe IO.
Un autre descendant de la classe IO est la classe Socket, qui peut être utilisée avec **Document.new** pour obtenir un fichier XML via une connexion réseau.

Si le constructeur Document prend en paramètre un objet Document, il sera intégralement cloné dans le nouvel objet Document. Si le constructeur prend en paramètre un objet String, la chaîne attendue devra contenir un flux XML. Petit exemple :

code2.rb - Affichage d'un XML contenu dans une chaîne

```
require 'rexml/document'
include REXML
string = <<EOF
<?xml version="1.0" encoding="ISO-8859-15"?>
<!DOCTYPE bibliography PUBLIC "-//OASIS//DTD DocBook
XML V4.2//EN"
    "http://www.oasis-
open.org/docbook/xml/4.2/docbookx.dtd">
<bibliography>
    <biblioentry id="FHIW13C-1234">
      <author>
        <firstname>Godfrey</firstname>
        <surname>Vesey</surname>
      </author>
      <title>Personal Identity: A Philosophical
Analysis</title>
      <publisher>
        <publishername>Cornell University
Press</publishername>
      </publisher>
      <pubdate>1977</pubdate>
   </biblioentry>
</bibliography>
EOF
doc = Document.new(string)
puts doc
```

Nous avons utilisé un document de type String : tout les caractères compris entre <<EOF et EOF, nouvelles lignes incluses, font partis de la chaîne.

2. Accéder aux éléments et aux attributs

A partir de maintenant, nous allons utiliser irb, le débuggeur interactif de Ruby, pour les exemples d'utilisation de la librairie REXML.
Au prompt d'irb, nous allons charger le fichier bibliography.xml dans un document. Après ça, nous pourrons exécuter les commandes pour accéder aux éléments et aux attributs de notre

document de manière interactive.

```
koan$ irb
irb(main):001:0> require 'rexml/document'
=> true
irb(main):002:0> include REXML
=> Object
irb(main):003:0> doc =
Document.new(File.new("bibliography.xml"))
=> <UNDEFINED> ... </>
```

Maintenant, nous pouvons explorer notre document très facilement. Jetons un oeil à une session irb typique avec notre fichier XML :

```
irb(main):004:0> root = doc.root
=> <bibliography id='personal_identity'> ... </>
irb(main):005:0> root.attributes['id']
=> "personal identity"
irb(main):006:0> puts
root.elements[1].elements["author"]
<author>
  <firstname>Godfrey</firstname>
  <surname>Vesey</surname>
</author>
irb(main):007:0> puts
root.elements["biblioentry[1]/author"]
<author>
  <firstname>Godfrey</firstname>
  <surname>Vesey</surname>
</author>
irb(main):008:0> puts
root.elements["biblioentry[@id='FHIW13C-1260']"]
<biblioentry id='FHIW13C-1260'>
      <author>
        <firstname>Sydney</firstname>
        <surname>Shoemaker</surname>
      </author>
      <author>
        <firstname>Richard</firstname>
        <surname>Swinburne</surname>
      </author>
      <title>Personal Identity</title>
      <publisher>
        <publishername>Basil Blackwell</publishername>
      </publisher>
      <pubdate>1984</pubdate>
    </biblioentry>
=> nil
irb(main):009:0> root.each_element('//author') {|
author| puts author}
<author>
  <firstname>Godfrey</firstname>
  <surname>Vesey</surname>
</author>
<author>
  <firstname>René</firstname>
  <surname>Marres</surname>
</author>
<author>
  <firstname>James</firstname>
  <surname>Baillie</surname>
</author>
<author>
  <firstname>Brian</firstname>
  <surname>Garrett</surname>
</author>
<author>
  <firstname>John</firstname>
  <surname>Perry</surname>
</author>
<author>
  <firstname>Geoffrey</firstname>
  <surname>Madell</surname>
</author>
<author>
  <firstname>Sydney</firstname>
  <surname>Shoemaker</surname>
</author>
<author>
  <firstname>Richard</firstname>
  <surname>Swinburne</surname>
</author>
<author>
  <firstname>Jonathan</firstname>
  <surname>Glover</surname>
</author>
<author>
  <firstname>Harold</firstname>
  <othername>W.</othername>
  <surname>Noonan</surname>
</author>
=> [<author> ... </>, <author> ...
  </>, <author> ... </>, <author> ...
  </>, <author> ... </>, <author> ...
  </>, <author> ... </>, <author> ...
  </>, <author> ... </>, <author> ... </>]
```

Premièrement, nous utilisons le nom "root" pour accéder à la racine de notre document. Ici, la racine du document est l'élément bibliography.
Chaque objet Element a un objet Attributs nommé "attributes" qui agit comme un tableau associatif avec le nom des attributs en guise de clé, et la valeur des attributs en guise de valeur.

Avec **root.attributes['id']** nous avons donc la valeur de l'attribut id de l'élément racine.
De la même façon, chaque objet Element contient un objet Element nommé "elements", et nous pouvons accéder aux sous-éléments en utilisant les méthodes each et [].

La méthode [] prend comme argument un index ou un Xpath, et retourne l'élément enfant qui correspond à l'expression.
Le Xpath fonctionne comme un filtre, qui va décider quels éléments doivent être retournés.
Notez que **root.elements[1]** est le premier élément enfant, car les index de Xpath commencent à 1, pas à 0. En fait, **root.elements[1]** est équivalent à **root.elements[*[1]]**, où *[1] est le Xpath du premier enfant.

La méthode each de la classe Element parcourt tous les éléments enfants, éventuellement en les filtrant suivant un Xpath donné. Le bloc de code sera alors exécuté à chaque itération. De plus, la méthode **Element.each_element** est un raccourci pour **Element.elements.each**.

3. Création et insertion d'éléments et d'attributs

Nous allons maintenant créer une petite bibliographie, consistant en une entrée unique. Voici comment elle se présente :

```
irb(main):010:0> doc2 = Document.new
=> <UNDEFINED/>
irb(main):011:0> doc2.add_element("bibliography",
                    {"id" => "philosophy"})
=> <bibliography id='philosophy'/>
irb(main):012:0> doc2.root.add_element("biblioentry")
=> <biblioentry/>
irb(main):013:0> biblioentry = doc2.root.elements[1]
=> <biblioentry/>
```

```
irb(main):014:0> author = Element.new("author")
=> <author/>
irb(main):015:0> author.add_element("firstname")
=> <firstname/>
irb(main):016:0> author.elements["firstname"].text =
"Bertrand"
=> "Bertrand"
irb(main):017:0> author.add_element("surname")
=> <surname/>
irb(main):018:0> author.elements["surname"].text =
"Russell"
=> "Russell"
irb(main):019:0> biblioentry.elements << author
=> <author> ... </>
irb(main):020:0> title = Element.new("title")
=> <title/>
irb(main):021:0> title.text = "The Problems of
Philosophy"
=> "The Problems of Philosophy"
irb(main):022:0> biblioentry.elements << title
=> <title> ... </>
irb(main):023:0> biblioentry.elements <<
Element.new("pubdate")
=> <pubdate/>
irb(main):024:0> biblioentry.elements["pubdate"].text =
"1912"
=> "1912"
irb(main):025:0> biblioentry.add_attribute("id",
"ISBN0-19-285423-2")
=> "ISBN0-19-285423-2"
irb(main):026:0> puts doc2
<bibliography id='philosophy'>
  <biblioentry id='ISBN0-19-285423-2'>
    <author>
      <firstname>Bertrand</firstname>
      <surname>Russell</surname>
    </author>
    <title>The Problems of Philosophy</title>
    <pubdate>1912</pubdate>
  </biblioentry>
</bibliography>
=> nil
```

Comme vous le voyez, nous créons un nouveau document vide dans lequel nous ajoutons un élément.

Cet élément devient l'élément racine (root). La méthode add_element prend le nom de l'élément en argument et un argument facultatif qui est la paire nom/valeur du tableau associatif de l'attribut.
Cette méthode ajoute donc un nouveau fils au document ou à l'élément, optionnellement elle peut aussi définir les attributs d'un élément.

Vous pouvez aussi créer un nouvel élément, comme nous l'avons fait avec l'élément "author", et l'ajouter après n'importe quel élément : si la méthode add_element prend un objet Element, celui-ci sera ajouté à l'élément parent.
A la place de la méthode add_element, vous pouvez aussi utiliser la méthode << sur Element.elements.
Ces deux méthodes retournent l'élément ajouté.

En complément, avec la méthode add_attribute, vous pouvez ajouter un attribut à un élément existant. Le premier paramètre est le nom de l'attribut, le second est sa valeur. La méthode retourne l'attribut qui a été ajoute.
La valeur du texte d'un élément peut être facilement changée avec **Element.text** ou bien avec la méthode add_text.

Si vous voulez insérer un élément à une position spécifique, vous pouvez utiliser les méthodes insert_before et insert_after :

```
irb(main):027:0> publisher = Element.new("publisher")
=> <publisher/>
irb(main):028:0> publishername =
Element.new("publishername")
=> <publishername/>
irb(main):029:0> publishername.add_text("Oxford
University Press")
=> <publishername> ... </>
irb(main):030:0> publisher << publishername
=> <publishername> ... </>
irb(main):031:0> doc2.root.insert_before("//pubdate",
publisher)
=> <bibliography id='philosophy'> ... </>
irb(main):032:0> puts doc2
<bibliography id='philosophy'>
  <biblioentry id='ISBN0-19-285423-2'>
    <author>
      <firstname>Bertrand</firstname>
      <surname>Russell</surname>
    </author>
    <title>The Problems of Philosophy</title>
    <publisher>
      <publishername>Oxford University
Press</publishername>
    </publisher>
    <pubdate>1912</pubdate>
  </biblioentry>
</bibliography>
=> nil
```

4. Suppression d'éléments et d'attributs

Les méthodes add_element et add_attribute ont leur équivalents respectifs pour détruire les éléments et les attributs. Voici comment cela fonctionne avec les attributs :

```
irb(main):033:0> doc2.root.delete_attribute('id')
=> <bibliography> ... </>
irb(main):034:0> puts doc2
<bibliography>
  <biblioentry id='ISBN0-19-285423-2'>
    <author>
      <firstname>Bertrand</firstname>
      <surname>Russell</surname>
    </author>
    <title>The Problems of Philosophy</title>
    <publisher>
      <publishername>Oxford University
Press</publishername>
    </publisher>
    <pubdate>1912</pubdate>
  </biblioentry>
</bibliography>
=> nil
```

La méthode delete_attribute retourne l'attribut détruit.

La méthode delete_element peut prendre un objet Element, une chaîne de caractères ou un index comme argument :

```
irb(main):034:0> doc2.delete_element("//publisher")
=> <publisher> ... </>
irb(main):035:0> puts doc2
<bibliography>
  <biblioentry id='ISBN0-19-285423-2'>
    <author>
      <firstname>Bertrand</firstname>
      <surname>Russell</surname>
    </author>
```

```
    <title>The Problems of Philosophy</title>
    <pubdate>1912</pubdate>
  </biblioentry>
</bibliography>
=> nil
irb(main):036:0> doc2.root.delete_element(1)
=> <biblioentry id='ISBN0-19-285423-2'> ... </>
irb(main):037:0> puts doc2
<bibliography/>
=> nil
```

Le premier appel a delete_element dans notre exemple utilise une expression XPath afin de localiser l'élément a détruire.
La seconde fois, nous utilisons l'index 1, ce qui signifie que le premier élément dans le document racine (root) sera détruit.
La méthode delete_element retourne l'élément détruit.

5. Noeud texte et traitement des entités

Nous avons déjà utilisé les noeuds texte dans les exemples précédents.

Dans cette section nous allons voir des fonctions avancées avec ces noeuds texte. Spécialement, Comment REXML prend en compte les entités ?

REXML n'est pas un parser validateur, et donc il n'est pas nécessaire d'assigner les entités externes. Les entités externes ne sont donc pas remplacées par leur valeur, mais les entités internes le sont: Quand REXML parcourt un document XML, il traite la DTD et crée une table avec les entités internes et leur valeur.
Lorsque l'une de ces entités est rencontrée dans le document, REXML la remplace par sa valeur.

Un exemple :

```
irb(main):038:0> doc3 = Document.new('<!DOCTYPE
testentity [
irb(main):039:1' <!ENTITY entity "test">]>
irb(main):040:1' <testentity>&entity; the
entity</testentity>')
=> <UNDEFINED> ... </>
irb(main):041:0> puts doc3
<!DOCTYPE testentity [
<!ENTITY entity "test">]>
<testentity>&entity; the entity</testentity>
=> nil
irb(main):042:0> doc3.root.text
=> "test the entity"
```

Vous pouvez voir que le document XML, lors de son impression, contient l'entité correcte. Lorsque vous accédez au texte, l'entité "&entity;" est correctement transformée en "test".

Cependant, REXML n'utilise pas une évaluation très poussée des entités. Comme résultat, nous voyons ce problème survenir :

```
irb(main):043:0> doc3.root.text = "test the &entity;"
=> "test the &entity;"
irb(main):044:0> puts doc3
<!DOCTYPE testentity [
<!ENTITY entity "test">
]>
<testentity>&entity; the &entity;</testentity>
=> nil
irb(main):045:0> doc3.root.text
=> "test the test"
```

Comme vous le voyez, le texte "test the &entity;" a été modifie en "&entity; the &entity;".

Si vous changez la valeur de l'entité, cela vous retournera un résultat différent de votre attente : plus de chose seront modifiées dans votre document que vous ne le vouliez.

Si cela est problématique pour votre application, vous pouvez appliquez le flag :raw sur n'importe quel noeud texte ou Elements, et même sur le noeud Document. Les entités dans ce noeud ne seront pas traitées et dans ce cas la, vous aurez à les traiter par vous même

Un exemple :

```
irb(main):046:0> doc3 = Document.new('<!DOCTYPE
testentity [
irb(main):047:1' <!ENTITY entity "test">]>
irb(main):048:1' <testentity>test the
&entity;</testentity>',
                 {:raw => :all})
=> <UNDEFINED> ... </>
irb(main):049:0> puts doc3
<!DOCTYPE testentity [
<!ENTITY entity "test">
]>
<testentity>test the &entity;</testentity>
=> nil
irb(main):050:0> doc3.root.text
=> "test the test"
```

Les caractères spéciaux comme "**&**", "**<**", "**>**", """ (guillemets), et ' sont automatiquement convertis.
D'ailleurs, si vous écrivez un de ces caractères dans un noeud texte ou dans un attribut, REXML les convertira dans son entité équivalente. Ex: "**&**" pour "**&**".

6. Parcours évenementiel

Le parcours évenementiel est plus rapide que le parcours d'arbre. Si la vitesse est un critère, le parcours évenementiel peut être utile.

Cependant, les options comme XPath ne sont pas valides. Vous devez avoir une class d'audit ("listener") et chaque fois que REXML rencontrera un évènement (balise de début, balise de fin, texte, etc.), le "listener" recevra une notification de l'évènement.

Un programme d'exemple :

code3.rb - Parcours évenementiel en action

```
require 'rexml/document'
require 'rexml/streamlistener'
include REXML

class Listener
  include StreamListener
  def tag_start(name, attributes)
    puts "Start #{name}"
  end
  def tag_end(name)
    puts "End #{name}"
  end
end

listener = Listener.new
parser =
Parsers::StreamParser.new(File.new("bibliography2.xml")
, listener)
parser.parse
```

bibliography2.xml : Lien44

Exécuter code3.rb donne cette sortie :

```
koan$ ruby code3.rb
Start bibliography
Start biblioentry
Start author
Start firstname
End firstname
Start surname
End surname
End author
Start title
End title
Start publisher
Start publishername
End publishername
End publisher
Start pubdate
End pubdate
End biblioentry
End bibliography
```

7. Conclusion

Ruby et XML font une bonne équipe.
Le processeur XML REXML vous permet de créer, accéder et modifier vos documents XML en une seule fois et ce de façon très intuitive. Avec l'aide du debugger interactif irb de Ruby, vous pouvez aussi lire vos documents XML très aisément.

Retrouvez l'article de Koen Vervloesen en ligne : Lien45

Compte rendu de la conférence "Ruby on Rails"

Le 8 février 2007 se tenait à La Défense la conférence Valtech Training "Ruby on Rails". Une très bonne organisation, un accueil chaleureux et un contenu très interessant, de quoi donner envie d'adopter cette belle technologie.

1. Programme de la matinée

Le programme de la matinée a été intense, et très riche. Les locaux étaient faciles à trouver, nous avons étés très bien accueillis, puis nous sommes passés aux choses sérieuses :

- Présentation de Ruby
- Débuter avec RoR
  - Introduction
  - Framework MVC
  - Principe DRY
  - Composants
  - Configuration minimale et règles de déduction
  - Rake
  - Environnements
  - DBs
  - Générateurs
  - Tests
- Les modules de RoR
  - Active Record
  - Action Mailer
  - Action Web Service
  - Ajax on Rails
- Conclusion
- Démonstration

Comme vous pouvez le constater, un programme bien dense pour une bien courte matinée.

1.1. Présentation de Ruby

Une courte introduction à Ruby, qui est tout de même la base de Rails. Du coup, quelques incompréhensions par la suite dans les divers exemples, mais il faut bien réussir à tout faire tenir dans le délai imparti...
Dommage tout de même de ne pas plus insister sur ces bases, d'autant que le langage présente tout de même certaines syntaxes particulières par rapport à Java ou PHP.

1.2. Débuter avec RoR

Cette présentation fut beaucoup plus complète que la précédente (ça tombe bien, c'était tout de même le thème de la conférence).

Tout les principes de RoR ont étés détaillés : la structure d'une application RoR, le framework avec une approche MVC très facile à apréhender, conventions plutôt que configuration, DRY (Don't Repeat Yourself) qui prône la réutilisation maximum du code, les différentes bases de données supportées, les outils mis à disposition pour les tests tant unitaires que fonctionnels, etc.

1.3. Les modules de RoR

C'est là que nous avons abordé ce qui distingue vraiment Rails des autres langages : ses différents modules, et plus particulièrement ActiveRecord.

Active Record est le nerf de la guerre, ce qui rend RoR magique...
Il suffit de plancher sur la conception de sa base de données, de la créer, puis de taper une simple commande. Rails va alors se charger de nous générer des formulaires gérant le CRUD (Create, Read, Update, Delete). En moins de temps qu'il n'en faut pour le dire, on se retrouve avec ne application fonctionnelle.
Rails va de plus se charger de créer les objets correspondant à vos tables ainsi qu'un tas de méthodes pour les manipuler plus facilement.

La validation des données devient également d'une simplicité enfantine, puisqu'une simple ligne permet de s'assurer que les données saisies rentrent dans le cadre défini, Rails gère le reste tout seul.

Le seul inconvénient pour le moment est que Rails ne gère pas les procédures stockées.

Action Mailer quant à lui permet de gérer l'envoi automatique de mail, ainsi que la réception et le traitement automatique. Il devient facile de déclencher une action lors de la réception d'un mail.

Action Web Service permet quand à lui de gérer les sevices web. Il supporte les protocoles SOAP et XML-RPC.
Encore une fois, dommage de ne pas avoir eu plus de temps à consacrer à cette partie, mais n'oublions pas que cette conférence n'était qu'un survol des possibilités de Rails, et non pas une formation ;-)

Au tour d'Ajax on Rails.
Vous vous en doutez, le but est d'intégrer Ajax dans Rails. L'idée est ici de ne coder qu'en Ruby, Rails se charge de faire l'interface

avec Prototype.js pour générer le code qui va bien.

1.4. Conclusion

La conclusion a été l'occasion de récapituler ce que nous avons vu avant, ainsi que de nous présenter sommairement les divers outils avec lesquels développer pour RoR : InstantRails, RadRails, Capistrano, les différents serveurs, etc.
Ce fut également un moment d'échange où nous avons pu poser les questions qui nous étaient venues à l'esprit pendant la présentation.

1.5. Démonstration

Un gros bravo pour cette partie !

L'intervenant à sous nos yeux réalisé une micro-application (une todo-list) avec Rails, ce qui a permis à tous de se rendre compte de la simplicité d'utilisation et de développement.

Certes, ce n'était qu'une petite application, mais l'utilité d'Active Record a largement été démontrée, ainsi que la simplicité d'utilisation du modèle MVC.
Ainsi, rien de plus simple que de modifier le formulaire pour passer d'un affichage anglais de la date à un affichage français.

Nous avons également eu le temps de voir comment gérer les évolutions de nos bases, comment gérer un retour à l'état précédent, la manière dont fonctionnent les tests, ...

Bref, ce fut le complément idéal à la présentation, la mise en application de tout ce qui nous avait été dit. Une grande réussite !

2.Conclusion

Je tiens à remercier Valtech Training pour son accueil ainsi que pour la qualité de la conférence.

Retrouvez le compte-rendu de Pierre-Baptiste Naigeon en ligne : Lien46

FAQ Ruby

Comment installer un Gem ?

Pour installer un Gem, il y a deux méthodes :

- Si le gem se trouve dans le registre des gems distants , vous devez exécuter :

```
gem install leNomDeMonGem
```

- Si vous avez déjà télécharger le fichier .gem, allez dans le répertoire où se trouve le .gem, et exécutez :

```
gem install leNomDeMon.gem
```

Qu'est ce qu'un Gem ?

Un Gem est un paquet fournit par la communauté de Ruby.
On peut comparer ça en quelque sorte a un "zip", il contient le module à installer, et un descripteur qui est utiliser pour l'installer (et lu automatiquement par RubyGems).
le Gem peut être téléchargé manuellement ou automatiquement via RubyGems.

RubyGems , c'est quoi ?

RubyGems est un gestionnaire de Package Similaire a "apt-get", mais qui est codé en "PureRuby", et sert a installer/supprimer/mettre à jour des modules Ruby.

Comment récupérer les arguments passés en ligne de commande ?

Il suffit d'utiliser ARGV. ARGV est un tableau contenant les arguments de la ligne de commande (après ruby xxx.rb) sous forme de chaîne et séparés par un espace.

Ex : ruby mon_script.rb coucou

```
puts ARGV[0]
```

Le code ci-dessus va nous renvoyer la chaîne "coucou". A noter que le tableau ARGV ne contiendra que des chaînes, il faudra donc veiller à les convertir suivant le cas.

Comment passer des arguments facultatifs à une méthode ?

Deux solutions :

- Définir une valeur par défaut :

```
def methode_test(x, y, z=42)
end

mon_objet.methode_test(12, 24)
```

Dans ce cas, z prendra comme valeur par défaut 42 s'il est omis dans l'appel (sinon, il prend la valeur passée dans l'appel).

- Utiliser un tableau de paramètres :
Si vous ne savez pas combien de paramètres vous voulez passer, vous pouvez utiliser la syntaxe suivante :

```
def methode_test2(x, y, *z)
end

mon_objet.methode_test2(12, 24, 42, 72, 128)
```

z sera alors un tableau contenant [42, 72, 128].

Retrouvez la FAQ Ruby en ligne : Lien47

Le Club

Ouverture de la boutique du club des développeurs

Suite à de nombreuses demandes de la part des membres du club, Developpez.com est fier de vous présenter la nouvelle Boutique du Club des Développeurs.

Vous y trouverez des T-Shirts à l'effigie de votre club, des tapis de souris, des tasses à café ainsi que des casquettes.

Tous les objets présents dans cette boutique sont vendus à prix coutant, Developpez.com ne réalise donc aucun bénéfice avec cette vente, c'est donc encore un nouveau service gratuit que nous vous proposons.

Accédez dès maintenant à la boutique : Lien48

Nouveaux records

Vous êtes toujours plus nombreux à accéder au site et au forum de Developpez.com. Cela se remarque par les nouveaux records de fréquentation que nous avons atteints.

Ainsi, vous étiez 2800 à être connecté en même temps sur le forum le 14 Février de cette année à 15h41.

Et le site de Developpez.com a reçu 106400 visites en une seule journée, le mardi 13 février.

Merci de votre fidélité.

Liens

Lien1 : http://ydisanto.ftp-developpez.com/tutoriels/j2se/runtime/fichiers/ProcessLauncher.java
Lien2 : http://ydisanto.developpez.com/tutoriels/j2se/runtime
Lien3 : http://java.sun.com/javase/6/docs/api/javax/swing/SwingWorker.html
Lien4 : http://rom.developpez.com/java-swingworker/
Lien5 : http://developer.mozilla.org/fr/docs/XULRunner
Lien6 : http://wiki.eclipse.org/index.php/Europa_Simultaneous_Release
Lien7 : http://blog.developpez.com/index.php?blog=124&title=eclipse_3_3_m5_est_disponible_et_europa
Lien8 : http://jcp.org/aboutJava/communityprocess/final/jsr270/index.html
Lien9 : http://www.javalobby.org/java/forums/t88549.html
Lien10 : http://blog.developpez.com/index.php?blog=51&p=2751&more=1
Lien11 : http://www.freebsd.org/cgi/man.cgi?query=login.conf
Lien12 : http://julp.developpez.com/freebsd/passerelle-authentification/authpf-admin.tar.gz
Lien13 : http://julp.developpez.com/freebsd/passerelle-authentification/
Lien14 : http://blog.developpez.com/index.php?blog=66&p=2842&more=1&c=1&tb=1&pb=1#more2842
Lien15 : http://linux.developpez.tv/2007/solutionslinux/
Lien16 : http://jc-cornic.developpez.com/tutoriels/php/pdf/fichiers/phpToPDF.zip
Lien17 : http://www.fpdf.org
Lien18 : http://jc-cornic.developpez.com/tutoriels/php/pdf/?page=page_4
Lien19 : http://iteratif.developpez.com/articles/as3/architecture/
Lien20 : http://msdn.microsoft.com/library/fre/default.asp?url=/library/fre/csspec/html/vclrfcsharpspec_17_4_1.asp
Lien21 : http://vincentlaine.developpez.com/tuto/dotnet/customattributs/
Lien22 : http://badger.developpez.com/tutoriels/dotnet/customfiletypes/
Lien23 : http://broux.developpez.com/articles/c/driver-c-linux/
Lien24 : http://c.developpez.com/livres/
Lien25 : http://grandfather.developpez.com/articles/openxml/structure/
Lien26 : http://idealliance.org/proceedings/xtech05/
Lien27 : http://2006.xtech.org/
Lien28 : http://xtech.expectnation.com/event/1
Lien29 : http://erwy.developpez.com/presentation/XTech
Lien30 : http://jab.developpez.com/tutoriels/db2/optimisation
Lien31 : http://baptiste-wicht.developpez.com/tutoriel/ms-sql/datetime
Lien32 : http://c.developpez.com/faq/cpp/
Lien33 : http://baptiste-wicht.developpez.com/tutoriel/reseau/introduction/
Lien34 : http://support.microsoft.com/kb/304387/en-us
Lien35 : http://office.microsoft.com/fr-fr/assistance/HP051876761036.asp
Lien36 : http://vb.developpez.com/sources/?page=word#fusion
Lien37 : http://silkyroad.developpez.com/Excel/PublipostageWordExcel
Lien38 : http://vb.developpez.com/faqvba/
Lien39 : http://laurent-dardenne.developpez.com/articles/Delphi/Interfaces/
Lien40 : news://forums.talkto.net:119/jedi.jcl
Lien41 : http://delphi.developpez.com/interviews/2006/florent-ouchet-jcl/
Lien42 : http://delphi.developpez.com/faq
Lien43 : http://ruby.developpez.com/cours/parser-xml-en-ruby/fichiers/bibliography.xml
Lien44 : http://ruby.developpez.com/cours/parser-xml-en-ruby/fichiers/bibliography2.xml
Lien45 : http://ruby.developpez.com/cours/parser-xml-en-ruby/
Lien46 : http://pbnaigeon.developpez.com/conference/ror-valtech-training
Lien47 : http://ruby.developpez.com/faq
Lien48 : http://boutique.developpez.com